ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA STROJNÍ

Studijní program:	N2301	Strojní inženýrství
Studijní obor:	2301T007	Průmyslové inženýrství a management

DIPLOMOVÁ PRÁCE

Nasazení Microsoft SQL Serveru pro databáze ve strojírenství

Autor: **Bc. Marek Kysela**

Vedoucí práce: **Ing. Petr Hořejší, Ph.D**

Akademický rok 2015/2016

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta strojní
Akademický rok: **2015/2016**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Marek KYSELA**
Osobní číslo:	**S14N0035P**
Studijní program:	**N2301 Strojní inženýrství**
Studijní obor:	**Průmyslové inženýrství a management**
Název tématu:	**Nasazení Microsoft SQL Serveru pro databáze ve strojírenství**
Zadávající katedra:	**Katedra průmyslového inženýrství a managementu**

Zásady pro vypracování:

1. Úvod
2. MS SQL Server - popis, možnosti, uživatelé, základní správa databází
3. Možnosti bezpečnosti a zálohování (různá řešení a jejich výhody a nevýhody)
4. Rozšířené služby (reporting, DB analysis)
5. Ukázka implementace SQL Serveru pro vybranou aplikaci
6. Závěr

Rozsah grafických prací: **0 výkresů**

Rozsah kvalifikační práce: **50 - 70 stran**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

1. *Books Online for SQL Server 2014,* **dostupné na www.mdsn.com**
2. **JORGENSEN A.** *Professional Microsoft SQL Server 2014 Administration.* **Wrox, 2014. ISBN 978-1118859131**
3. **MISTRY R., MISNER S.** *Introducing Microsoft SQL Server 2014: Technical Overview.* **Microsoft Press, 2014. ISBN 978-0-7356-8475-1**
4. **MAREŠ, J.** *Podnikové informační systémy a DP,* **e book. Plzeň: SmartMotion, 2012. ISBN 978-80-87539-05-7**

Vedoucí diplomové práce: **Ing. Petr Hořejší, Ph.D.**
Katedra průmyslového inženýrství a managementu

Konzultant diplomové práce: **Doc. Ing. Zdeněk Ulrych, Ph.D.**
Katedra průmyslového inženýrství a managementu

Datum zadání diplomové práce: **21. září 2015**

Termín odevzdání diplomové práce: **20. května 2016**

Doc. Ing. Milan Edl, Ph.D.
děkan

Západočeská univerzita v Plzni
L.S.
Fakulta strojní

Doc. Ing. Michal Šimon, Ph.D.
vedoucí katedry

V Plzni dne 21. září 2015

Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě bakalářskou/diplomovou práci, zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů, uvedených v seznamu, který je součástí této bakalářské/diplomové práce.

V Plzni dne: ……. **. . . . . .**

podpis autora

Poděkování

Chtěl bych poděkovat všem z katedry průmyslového inženýrství a managementu za jejich trpělivost, pomoc a užitečné rady nezbytné k dokončení mé diplomové práce. Dále své rodině za podporu při studiu a také bohům za příležitosti, jež mi byly v životě nabídnuty.

ANOTAČNÍ LIST DIPLOMOVÉ PRÁCE

AUTOR	**Příjmení** Kysela	**Jméno** Marek
STUDIJNÍ OBOR	**2301T007 Průmyslové inženýrství a management**	
VEDOUCÍ PRÁCE	**Příjmení (včetně titulů)** Ing. Hořejší, Ph.D	**Jméno** Petr
PRACOVIŠTĚ	ZČU - FST - KPV	

DRUH PRÁCE	**DIPLOMOVÁ**	~~**BAKALÁŘSKÁ**~~	**Nehodící se škrtněte**
NÁZEV PRÁCE	**Nasazení Microsoft SQL Serveru pro databáze ve strojírenství**		

FAKULTA	strojní		**KATEDRA**	KPV		**ROK ODEVZD.**	2016

POČET STRAN (A4 a ekvivalentů A4)

CELKEM	68		**TEXTOVÁ ČÁST**	48		**GRAFICKÁ ČÁST**	20

STRUČNÝ POPIS (MAX 10 ŘÁDEK) **ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY**	Diplomová práce popisuje implementaci MS SQL Serveru pro vytvořenou aplikaci. V teoretické části jsou popsány jednotlivé komponenty MS SQL Serveru, dále možnosti uživatelů, možnosti zabezpečení a zálohování. V praktické části jsou poznatky z teoretické části aplikované na vytvořenou aplikaci.
KLÍČOVÁ SLOVA **ZPRAVIDLA JEDNOSLOVNÉ POJMY, KTERÉ VYSTIHUJÍ PODSTATU PRÁCE**	SQL, server, databáze, data, zálohování, WPF, entity framework

SUMMARY OF DIPLOMA SHEET

AUTHOR	**Surname** Kysela	**Name** Marek	
FIELD OF STUDY	**2301T007** " Industrial Engineering and Management"		
SUPERVISOR	**Surname (Inclusive of Degrees)** Ing. Hořejší, Ph.D	**Name** Petr	
INSTITUTION	ZČU - FST - KPV		
TYPE OF WORK	**DIPLOMA**	~~**BACHELOR**~~	**Delete when not applicable**
TITLE OF THE WORK	**Implementation of Microsoft SQL Server for Industrial Engineering Databases**		

FACULTY	Mechanical Engineering		**DEPARTMENT**	KPV		**SUBMITTED IN**	2016

NUMBER OF PAGES (A4 and eq. A4)

TOTALLY	68		**TEXT PART**	48		**GRAPHICAL PART**	20

BRIEF DESCRIPTION **TOPIC, GOAL, RESULTS AND CONTRIBUTIONS**	This thesis dissert on implementation of Microsoft SQL Server for created application. Main components of MS SQL Server are described in the theoretical part. Also there is a theory about users, security and backups. Those theoretical findings are aplicated for created application.
KEY WORDS	SQL, server, database, data, backup, WPF, entity framework

Obsah

1 Úvod

Každý strojírenský podnik v dnešní době pracuje s obrovským objemem dat. Tato data již nelze mít uložena v papírové formě nebo na jedné počítačové stanici izolované od ostatních. S daty je potřeba efektivně pracovat a mít je neustále k dispozici. Ve většině případů se s určitými daty musí pracovat na více počítačích, často ve stejný čas a nezávisle na sobě.

Za tímto účelem se využívá systémů s možností víceuživatelského přístupu k datům. V podniku se ve většině případů nachází počítač s databází (server), ke kterému se lze připojit z jiných počítačů podniku, nahlížet na data, měnit je, případně je mazat. Jelikož má k datům v databázi přístup více uživatelů najednou, je třeba vyřešit případné kolize mezi uživateli a zajistit, aby měl každý k dispozici aktuální data.

Dále je nutno vyřešit otázku zálohování dat, protože je v dnešní době stále většina systémů náchylných na nečekané situace typu výpadku proudu nebo v krajním případně nějaké živelné katastrofy. Dále hrozí poškození lidským faktorem, ať už úmyslné či neúmyslné. Tento problém vyvolává další otázku a to je otázka zabezpečení databáze.

V této práci je přístup ke cvičné databázi a práce s touto databází zajištěna pomocí softwaru Microsoft SQL Server, dále jsou popsány jednotlivé komponenty softwaru a ukázány možnosti nastavení databáze a možnosti řešení výše zmíněných úskalí.

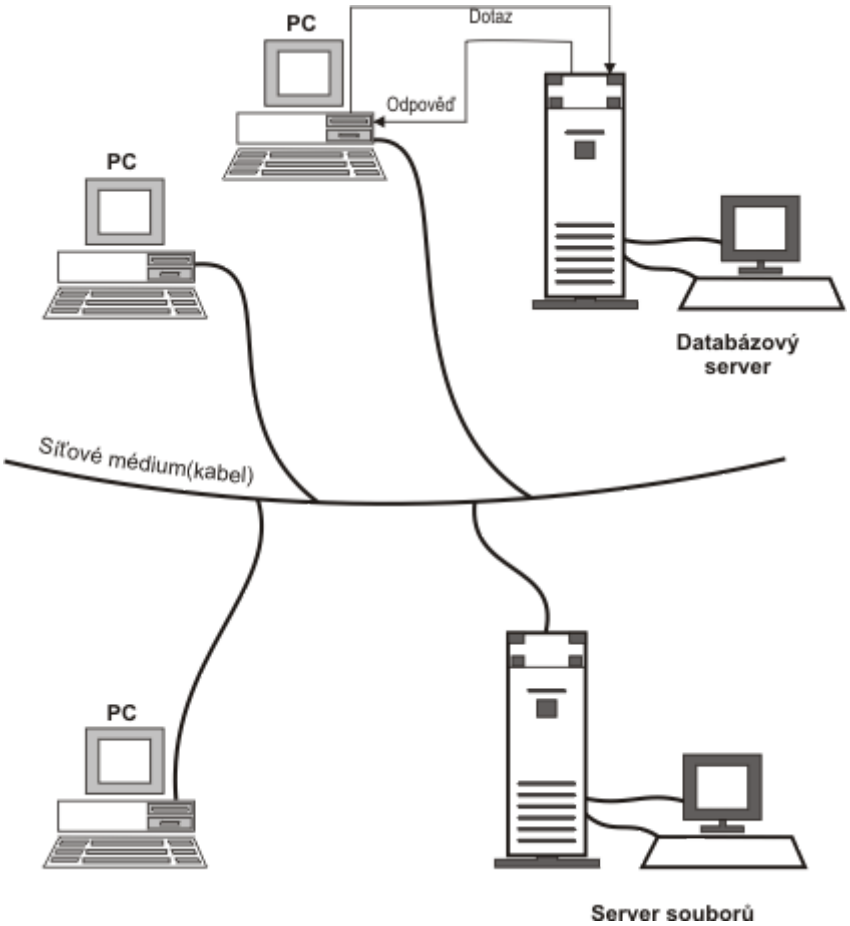
2 Základní pojmy

V této kapitole jsou vysvětleny základní pojmy vybraného tématu diplomové práce.

2.1 Server

V rámci výpočetní techniky je server program nebo zařízení poskytující určitou službu klientům (procesům či programům, které ji využívají). Této vazbě se říká model **klient-server**. Servery mohou poskytovat různé služby, například sdílení dat či zdrojů mezi více klienty nebo pro klienta provádět výpočetní úkony. Výpočetní kapacita serveru je rozdělena mezi jednotlivé procesy nebo zařízení. Proces klienta může běžet na stejném zařízení nebo se může k serveru připojit s využitím vzdáleného připojení z jiného zařízení. [1]

Systémy klient-server jsou v dnešní době nejčastěji implementovány jako modely **požadavek-odpověď** (dotaz-odpověď). Klient odešle požadavek serveru, který provede určitou akci a odešle klientovi odpověď. Nejčastěji s určitým výsledkem nebo potvrzením. Schéma systému klient-server je vidět na obrázku 2-1, kde klienty reprezentují 3 počítačové stanice. [1]



Obrázek 2-1 - Schéma systému klient-server [2]

Mezi nejběžnější servery patří databázový server, server souborů (file server), mail server, webový server, herní server nebo aplikační server. [3]

Z hlediska této práce je nejpodstatnější **databázový server**. Ten představuje soubor softwarových prostředků jednak pro práci s daty, ale i pro organizování a realizaci přístupu klientů k těmto údajům. [4]

2.2 Databáze

Databáze (neboli datová základna) je určitá uspořádaná množina dat. Zároveň tento pojem také zastřešuje nástroje pro ukládání a manipulaci s těmito daty. [4]
Databáze mají za úkol modelovat určitý aspekt reálného světa. Tuto část světa nazýváme **prostor problému**. Všechny tyto prostory jsou komplikované. Nejdůležitější je omezení navrhovaného databázového systému na konkrétní, dobře definovanou množinu objektů a jejich vzájemných vztahů. [5]

Datový model je myšlenkový popis prostoru problému. Patří sem definice entit, jejich atributů (např. entita – uživatel, atribut – jméno) a omezení entity. Datový model také vyjadřuje nejen vztahy mezi entitami, ale také omezení platná pro tyto entity. Ale datový model ještě nemá žádnou souvislost s fyzickým vzhledem výsledného systému. [5]

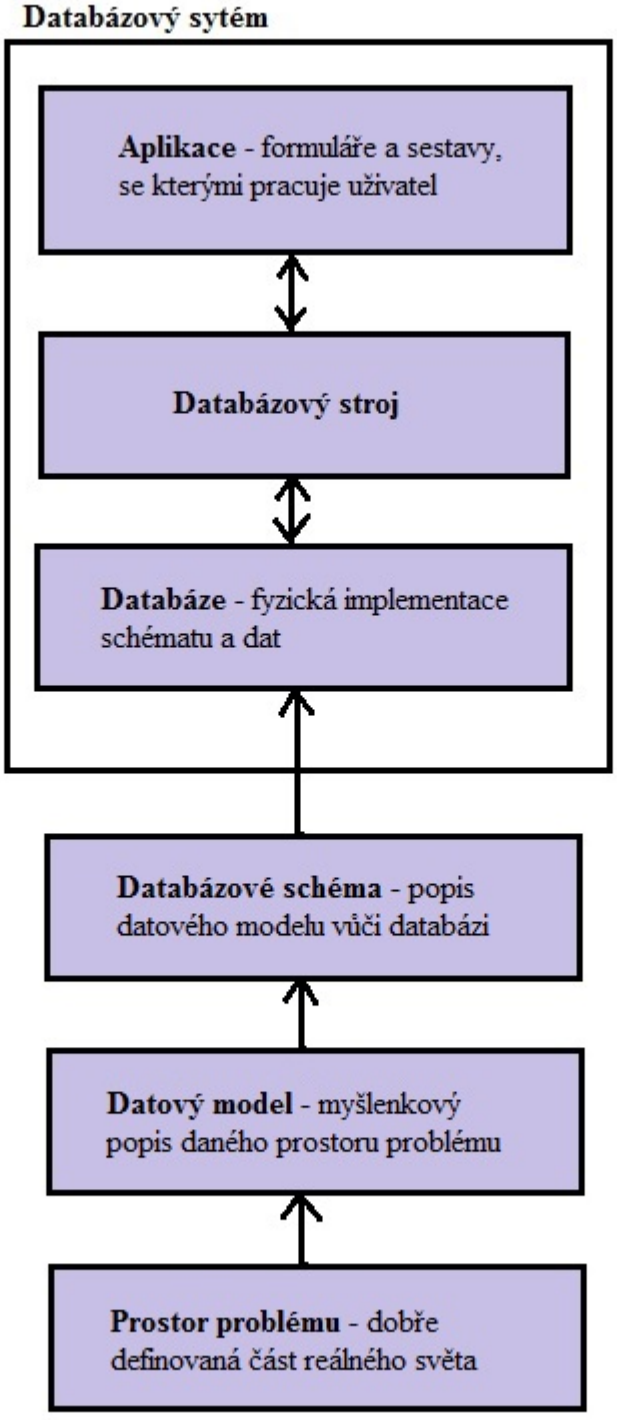
Databázové schéma označuje fyzické rozvržení systému, tedy seznam tabulek a pohledů. Je to také kolekce databázových objektů. Do této kolekce patří např. tabulky, uživatelé, procedury, triggery, sekvence, apod. Některé z nich patří do pokročilých znalostí o jazyku SQL. Databázová schémata vytváří správce SQL Serveru. Obecně každý uživatel má k dispozici svoje databázové schéma. To je defaultní, a proto všechny příkazy ohledně tabulek fungují. Nicméně se může stát, že vám správce serveru nastaví oprávnění manipulovat s tabulkami, které náleží jinému schématu. Takové tabulky musíme adresovat kromě jejich jména i jménem schématu, ve kterém se nachází. [5]

Databázové stroje představují mechanismy, které provádějí vlastní fyzickou manipulaci s daty, např. ukládání na disk, otevírání souborů a podobně. Konkrétní databázové stroje jsou třeba Microsoft Jet a SQL Server. Ty jsou sice odlišné, ale mají několik společných vlastností. Jsou to vynikající nástroje pro ukládání dat a práci s nimi. Rozdíl je v architektuře obou nástrojů a v typech problémů, které dokážou řešit. Microsoft Jet je typem spíše pro domácí uživatele, hodí se nejlépe pro menší nebo středně velké systémy. SQL Server je vhodný pro užívání ve velkých a rozsáhlých systémů a je přístupný stovkám uživatelů. Pracuje v architektuře klient-server. Uživatelé mohou pracovat současně s kriticky důležitými aplikacemi.[5]

Po vysvětlení, jak mají data podle naší představy vypadat, vytvoří stroj určité fyzické objekty, do nichž se poté ukládají data. Naše představa dat se vkládá buď pomocí programového kódu, nebo v grafickém prostředí, které má např. Microsoft Access. Databázi v této fázi tvoří tabulky, definované pohledy, dotazy, procedury a pravidla, jejichž dodržováním se docílí vyžadovaná úroveň ochrany dat. [5]

Databázový systém se tedy skládá z aplikace, databáze, databázového stroje a vrstvy middleware. Součástí je tedy všechen software a data, která tvoří reálný, provozní systém. Je tedy třeba si uvědomit, že pojem databáze nezahrnuje samotnou aplikaci, skládající se z formulářů a sestav, nebo middleware. Neobsahuje ani databázový stroj. [5]

Vztahy mezi výše zmíněnými pojmy jsou zobrazeny na následujícím obrázku 2-2.



Obrázek 2-2 - Vztahy mezi definovanými pojmy

2.3 SQL

SQL - Structured Query Language (Strukturovaný dotazovací jazyk) - je obecný nástroj pro manipulaci, správu a organizování dat uložených v databázi počítače. Je v prvé řadě určen uživatelům, i když jej v mnoha směrech využívají i tvůrci aplikací. Je adaptovatelný pro jakékoliv prostředí. [6]

Název tohoto nástroje je sice stručný, není však výstižný. SQL totiž není pouze dotazovací jazyk, s jeho pomocí je možno také definovat data, tedy strukturu tabulky, naplňovat sloupce tabulky (pole záznamů) daty a definovat vztahy a organizaci mezi položkami dat. Dále umožňuje řízení přístupu k datům, tedy udělování a odebírání přístupových oprávnění na různých úrovních, čímž chrání data před náhodným nebo úmyslným zničením, neautorizovaným čtením nebo manipulaci s nimi, dále také umožňuje sdílené využívání dat a zajišťuje hladký průběh činností, přistupuje-li k datům více uživatelů současně. [6]

SQL je specializovaný programovací jazyk, který se používá ve vhodném prostředí buď uživatelsky nebo interaktivně k okamžitému řešení úloh (nejčastěji dotazy), nebo se jeho příkazy vkládají do hostitelského jazyka. SQL není však plnohodnotným samostatným programovacím jazykem, např. proto, že se v něm ve většině implementací nenachází řídicí programové konstrukce a další požadované prvky, které by měl obsahovat každý obecný programovací jazyk. SQL je tedy standardizovaný nástroj pro práci s relačními databázemi. Nepředstavuje databázový systém, ale různě integrovanou součást systému řízení bází dat. [6]

SQL je především interaktivní dotazovací jazyk - umožňuje získat odpovědi i na velmi komplikované dotazy téměř ihned. Je nástrojem neprocedurálním, s množinovým přístupem k datům a je jazykem standardizovaným, je srozumitelný, protože chápe data v podobě tabulek, což je snadno pochopitelné i uživatelům. Pracuje s relačními databázemi, ve kterých se uživatel dívá na data v podobě soustavy provázaných tabulek. Každá tabulka představuje množinu dat, která je uspořádaná v řádcích (záznamech) a sloupcích (položkách). Na hodnotu dat se uživatel odkazuje jako na prvek v matici. V převážné většině případů je výsledkem úlohy popsané v SQL nějaká množina dat z jedné nebo více tabulek, tzv. tabulka výsledků, která nemusí být vždy konečným produktem. Může sloužit jako množina vstupních údajů pro další zpracování. Např. pro výtisk etiket nebo vykreslení grafu, atd. [6]

SQL může sloužit jako "společná řeč", zejména při provozu v sítích, na kterých se používají různé databázové produkty. Jazyk SQL se používá také k vytváření náhledů (dotazů). Náhledy SQL umožňují vytvořit pro různé uživatele různé pohledy na struktury tabulek a na data. Každý uživatel tak vidí pouze ta data, která má vidět. Přitom data vidí uživatel opět v podobě jednoduché tabulky, i když ve skutečnosti data pocházejí z různých tabulek. Data zobrazovaná v náhledech jsou dynamická, tzn. znění-li se data v tabulkách (databázových souborech), změní se také data, které zobrazuje náhled. Také naopak. Pracuje-li se s aktualizačním náhledem (speciální typ náhledu, který umožňuje nejen data prohlížet, ale i přidávat a aktualizovat) a změní se v něm data, promítnou se změny do patřičných tabulek (databázových souborů). [6]

SQL příkazy
SQL příkazy lze rozdělit do následujících podmnožin:

Data Definition Language (DDL) – pomocí těchto příkazů lze definovat, vytvářet, měnit a rušit (odstraňovat) různé objekty a struktury v relačních databázích, jako jsou například tabulky, indexy, spouště, uložené procedury a podobně. Taktéž lze přidělovat a odebírat uživatelská oprávnění jednotlivým uživatelům a skupinám uživatelů. Do této skupiny patří tyto příkazy: CREATE, ALTER, DROP.

Data Manipulation Language (DML) – do této podmnožiny patří příkazy pro manipulaci s daty neboli příkazy pro vkládání (INSERT), aktualizaci (UPDATE), mazání (DELETE) a výběr (SELECT) údajů.

Data Control Language (DLC) – speciální příkazy pro řízení provozu a údržby databáze. Patří sem příkazy GRANT, ALTER USER, DROP USER, REVOKE.

Transaction Control Commands – příkazy pro řízení transakcí, patří sem příkazy CREATE TRANSACTION a COMMIT.

3 Microsoft SQL Server

V této kapitole se nachází základní informace o systému Microsoft SQL Server (dále jen MS SQL Server), popis samotného systému, uživatelů a základní správy databází.

3.1 Popis

Microsoft SQL Server je relační databázový a analytický systém pro e-obchody, byznys a řešení datových skladů vyvinutý společností Microsoft. Je to nástroj pro správu služeb spojených s SQL Serverem ke konfiguraci síťových protokolů používaných SQL Serverem a ke správě konfigurace síťové konektivity z počítačů klientů SQL Serveru. [7]

Jedná se o nástroj, který zachycuje události ze serveru. Ty ukládá do souboru, který může být později analyzován nebo použit k zopakování určité řady kroků, když se snaží diagnostikovat daný problém. [7]

Slouží k činnostem, jako jsou:

- Krokování skrze problémy dotazů k nalezení příčiny problému.
- Hledání a diagnostikování pomalu běžících dotazů.
- Snímání série Transact-SQL příkazů, které vedou k problému. Uložené stopy pak mohou být vloženy na test server, kde může být problém diagnostikován.
- Monitorování výkonu SQL Serveru k ladění úloh. Pro informace o ladění designu fyzických databází pro databázi úloh.
- Korelace čítačů výkonu k diagnostikování problémů. [7]

Počítač, na který se bude instalovat MS SQL Server, se stane databázovým serverem. Na tomto zařízení může běžet nezávisle na sobě více databázových systémů, ty se v případě MS SQL Serveru nazývají instance databázového systému. Tyto instance se vzájemně neovlivňují, na každé instanci může být hostován libovolný počet databází.

Databázový systém má za úkol poskytnout rozhraní, přes které se připojuje klientská aplikace. To umožňuje, aby systém mohl posílat příkazy, jež se mají proti databázi vykonat. Způsobů navázání komunikace v SQL Serveru jsou tyto čtyři:

Sdílená paměť (shared memory) – přístup přes sdílenou paměť je bezesporu nejrychlejším nejméně problémovým způsobem komunikace. Nevyžaduje žádné nastavování a bývá implicitně povolen. Jeho nevýhodou je fakt, že funguje jen v rámci jednoho počítače – klient tedy musí být umístěn na počítači, kde běží databázový systém.

Pojmenované roury (named pipes) – další způsob mezi-procesové komunikace. Tentokrát však pro komunikaci jak v rámci jednoho počítače, tak v rámci lokální sítě. U tohoto způsobu komunikace se navíc nastavuje textové jméno komunikační roury, což je při tomto způsobu komunikace vyžadováno.

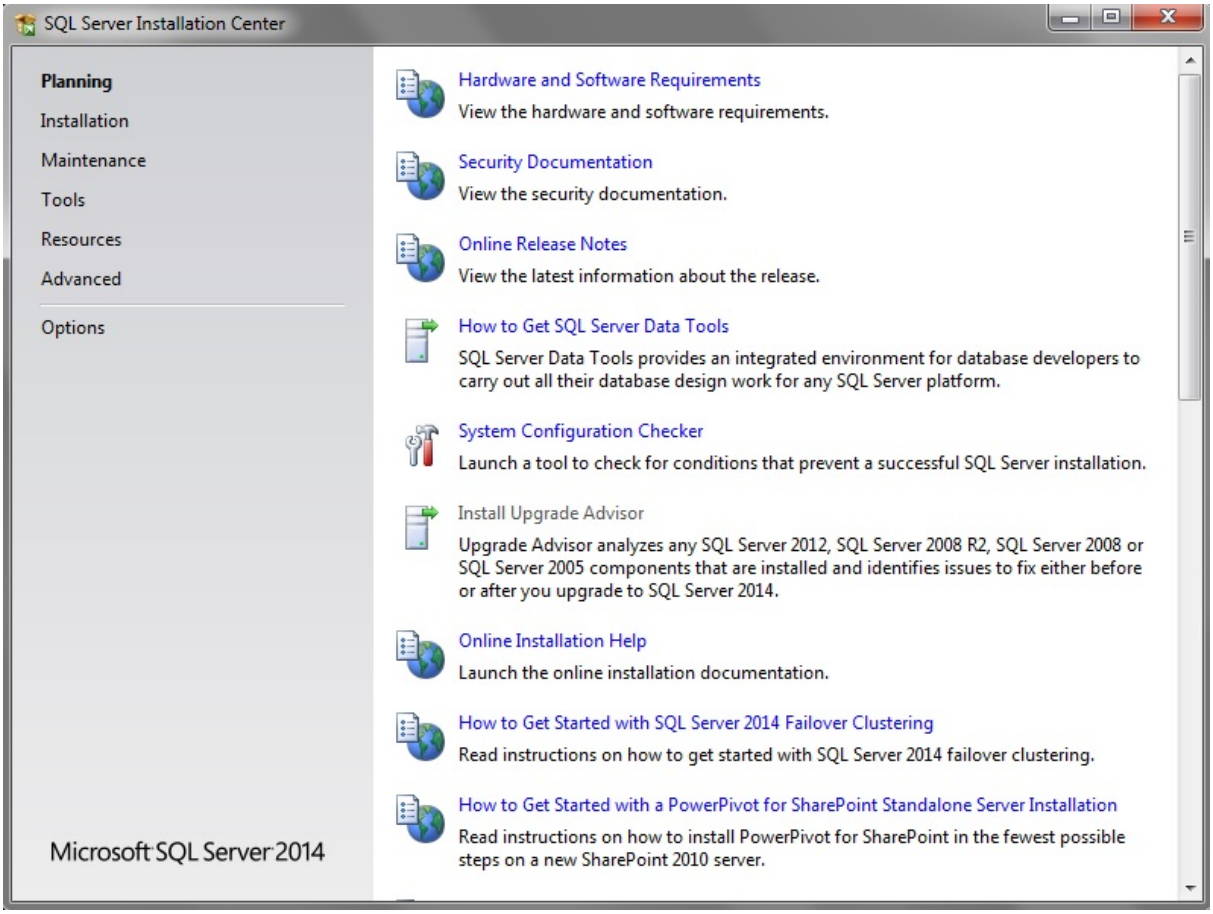
TCP/IP – Nejpoužívanější a oficiálně nejčastěji doporučovaný protokol. Při správné konfiguraci pracuje v rámci stejného počítače, místní sítě i internetu. U TCP/IP již narážíme na komplexnější konfiguraci.

3.2 Nástroje a komponenty MS SQL Server

Systém MS SQL Server je balíkem jednotlivých nástrojů a komponent, nejdůležitější z nich budou popsány v následující kapitole.

3.2.1 SQL Server Installation Center

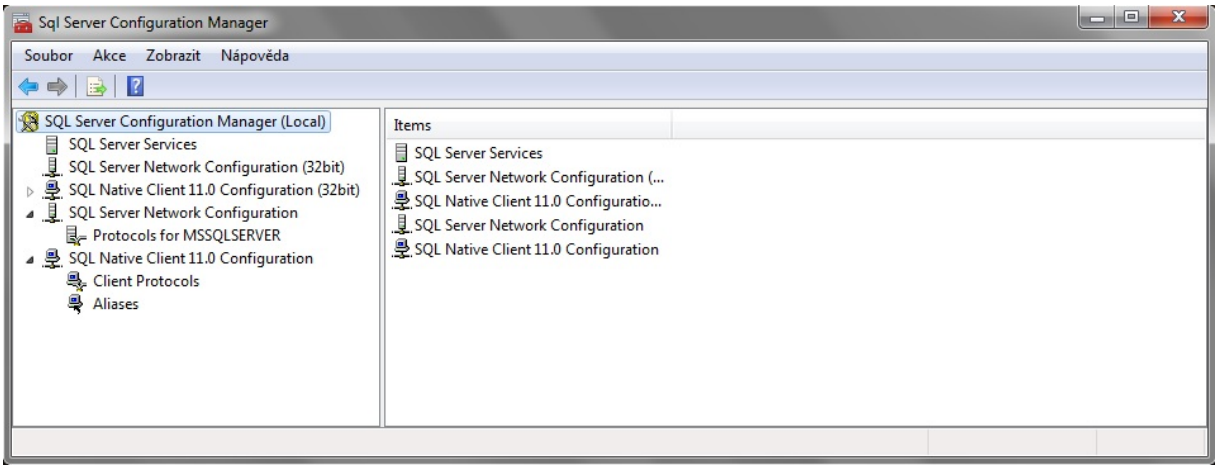
Jedná se o pomocný program přidaný do MS SQL Serveru, který má za úkol pomoct se spouštěním instalačních procesů. Zároveň poskytuje odkazy na dokumentaci a nástroje pro zjištění instalovaných instancí a komponent, či na kontrolu splnění systémových požadavků. Dále tento program nabízí možnosti údržby systému MS SQL Serveru. Funkce se hodí především při první instalaci jako analytický nástroj, zda je připraveno vše, co je potřeba a zda je již nainstalovaná nějaká instance. SQL Server Installation Center je vidět na následujícím obrázku 3-1.



Obrázek 3-1 - SQL Server Installation Center

3.2.2 SQL Configuration Manager

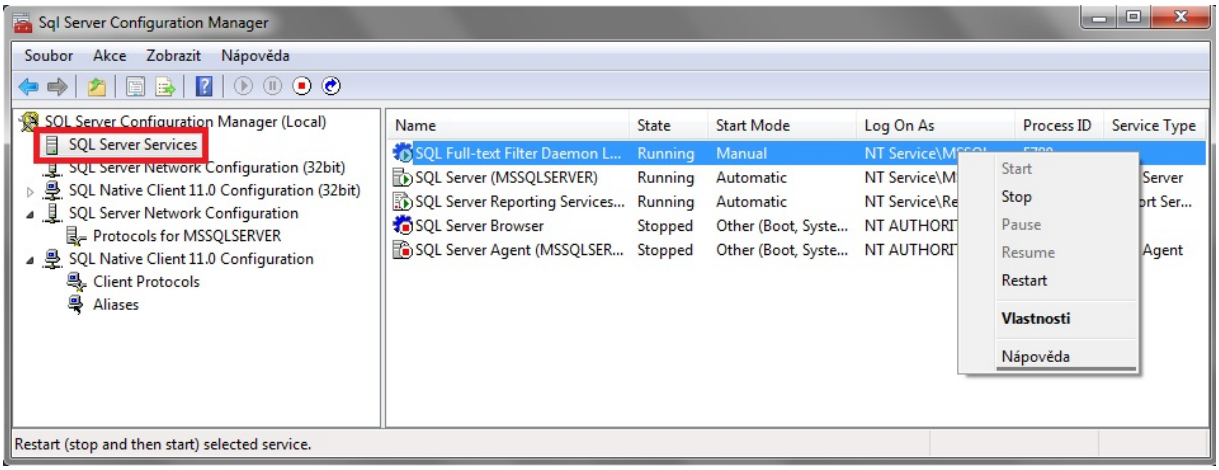
Tento nástroj umožňuje prohlížení již nainstalovaných instancí, nastavení a kontrolu parametrů všech služeb. Dále umožňuje konfiguraci způsobu komunikace databázové služby s klientem. SQL Configuration Manager je vidět na obrázku 3-2. Jednotlivé položky z levého sloupce jsou dále popsány dopodrobna.



Obrázek 3-2 - SQL Server Configuration Manager

SQL Server Services

Přehled služeb týkajících se SQL Serveru. Vhodné pro kontrolu, které služby běží, pro nastavování parametrů (vlastností) služeb. Dále je zde možnost služby restartovat, zastavovat a spouštět. Přehled služeb je vidět na obrázku 3-3. Jednotlivé služby jsou popsány podrobněji dále.



Obrázek 3-3 - SQL Sever Services

Služba SQL Full-text Filter Deamon Launcher (jméno_instance) - tato služba se stará o indexaci obsahu databáze pro fulltextové vyhledávání. Pokud není fulltext instalovaný, služba se zde nezobrazí. Pokud není služba spuštěna, fulltextové vyhledávání bude sice fungovat, ale nebude aktualizovat index. U této služby je vhodné v nastavení vlastností změnit manuální spouštění na spouštění automatické.

Služba SQL Server (jméno_instance) - jedná se o hlavní službu konkrétní instance SQL Serveru. Řeší jak přístup k datovým souborům, transakce, vykonávání a optimalizaci dotazů, tak komunikaci s klienty. Tato služba jako jediná musí být naprosto nezbytně spuštěná k fungování konkrétní instance SQL Serveru.

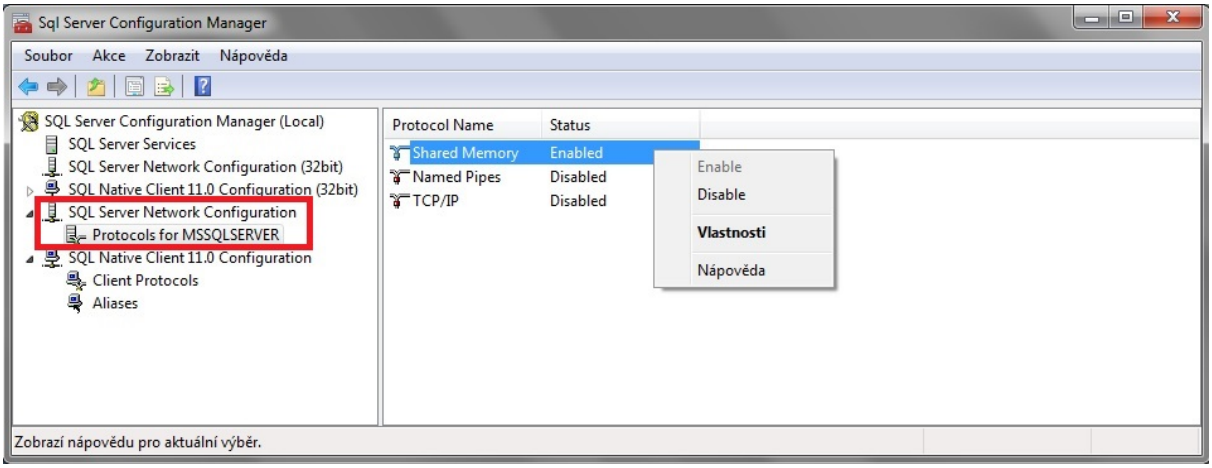
Služba SQL Reporting Services (jméno_instance) – tato služba obsahuje celou řadu nástrojů a služeb usnadňujících vytváření, nasazení a správu sestav. Díky tomu můžete z relačních či multidimenzionálních zdrojů dat nebo zdrojů dat založených na standardu XML vytvořit interaktivní, tabulkové nebo grafické sestavy či sestavy s volným tvarem. Je také možné publikovat sestavy, sestavy plánu zpracování, nebo přístup zprávy na vyžádání.

Služba SQL Server Browser – tato služba je společná pro všechny instance. Má za úkol dávat jednotně vědět klientům o instancích na serveru a způsobech jak se k nim připojit. Účelem je možnost se připojovat na instance bez nutnosti znát přesné parametry připojení (protokol, port a podobně). Stačí vědět pouze adresu serveru a jméno instance.

Služba SQL Server Agent (jméno_instance) – tato služba má za úkol spouštět plánované úlohy a obecně automatizované úlohy SQL Serveru.

SQL Server Network Configuration

Zde nalezneme tolik pododdílů, kolik máme nainstalovaných instancí. Pro každou z nich je určen jeden pododdíl Protocols for "jméno instance". V každém z těchto pododdílu nalezneme konfiguraci všech popsaných komunikačních protokolů. V kontextovém menu u každého z těchto protokolů lze nastavit, zda je povolen (Enabled), či zakázán (Disabled). Také je možné zobrazit vlastnosti a případně upravit konfiguraci konkrétního protokolu. Dané možnosti jsou vidět na obrázku 3-4.



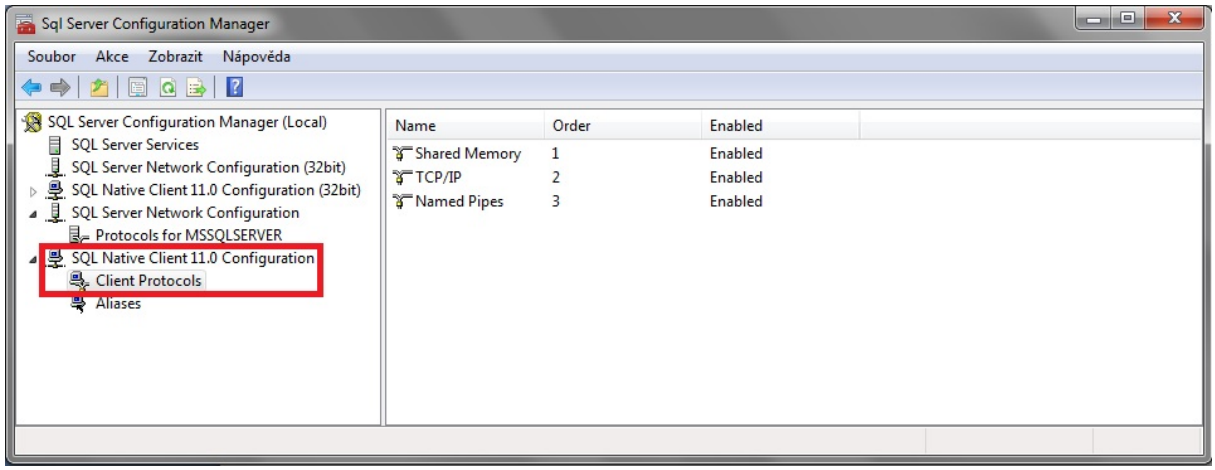
Obrázek 3-4 - SQL Server Network Configuration

SQL Native Client Configuration

Client Protocols - Tento oddíl nesouvisí s nastavením serveru. Definuje, v jakém pořadí se mají vyzkoušet protokoly pro připojení k serveru, pokud je tato stanice klientem. Pokud tedy například zde zakážeme TCP/IP, nikdy se z tohoto počítače nepřipojíme k žádnému SQL Serveru pomocí tohoto protokolu. Defaultní nastavení zkouší připojení v tomto logickém pořadí:

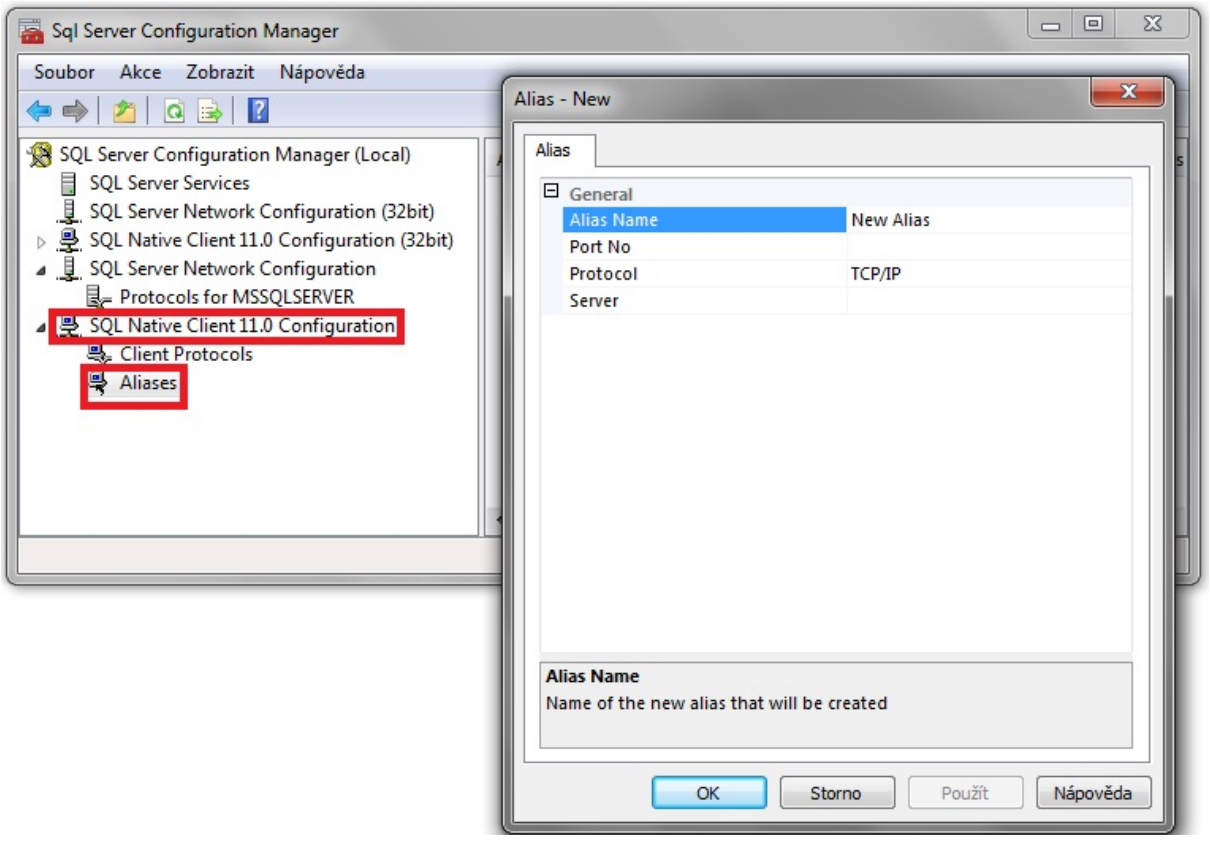
Shared memory – nejrychlejší připojení, pokud nelze využít Shared memory, tak TCP/IP na defaultním portu 1433, pokud nelze využít ani TCP/IP, tak Named Pipes.

Toto pořadí je vidět na obrázku 3-5. Je takto přednastaveno a většinou se nemění.



Obrázek 3-5 - Client Protocols

Aliases – v praxi může nastat problém, kdy je aplikace nastavena na komunikaci s konkrétním serverem a server buď již neexistuje, nebo jej chceme dočasně změnit. Pokud je změna konfigurace připojovacích adres složitá, lze použít právě SQL Server Alias. Tím definujeme: pokud se kterákoliv aplikace připojující se z tohoto počítače pokusí připojit na server uvedený jako jméno aliasu (Alias name), komunikace se místo toho přesměruje na adresu serveru (Server) pod zvoleným prokolem (Protocol) a portem (Port No). Přidání nového aliasu je vidět na obrázku 3-6.



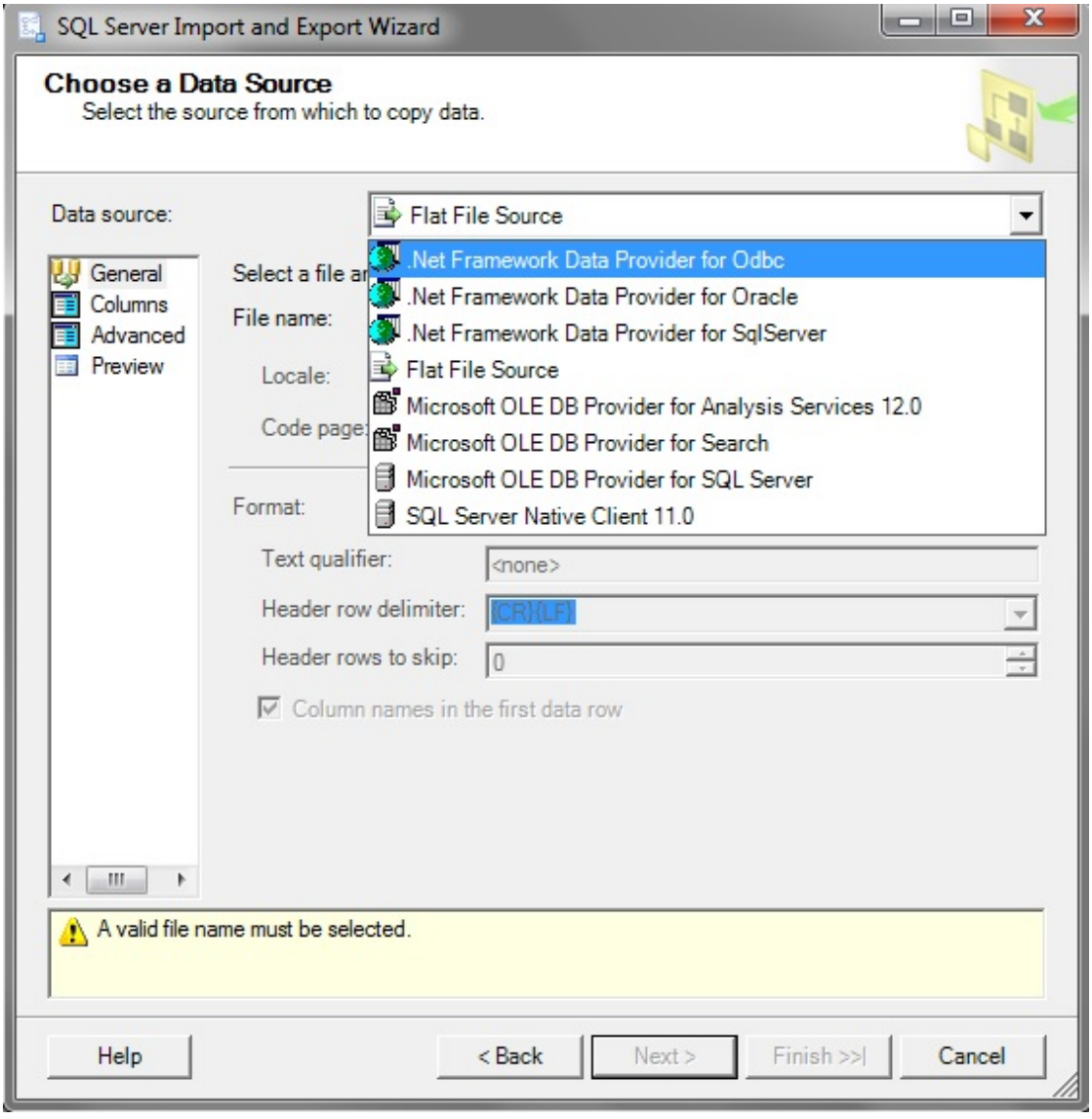
Obrázek 3-6 - Přidání nového aliasu

3.2.3 SQL Import and Export Data

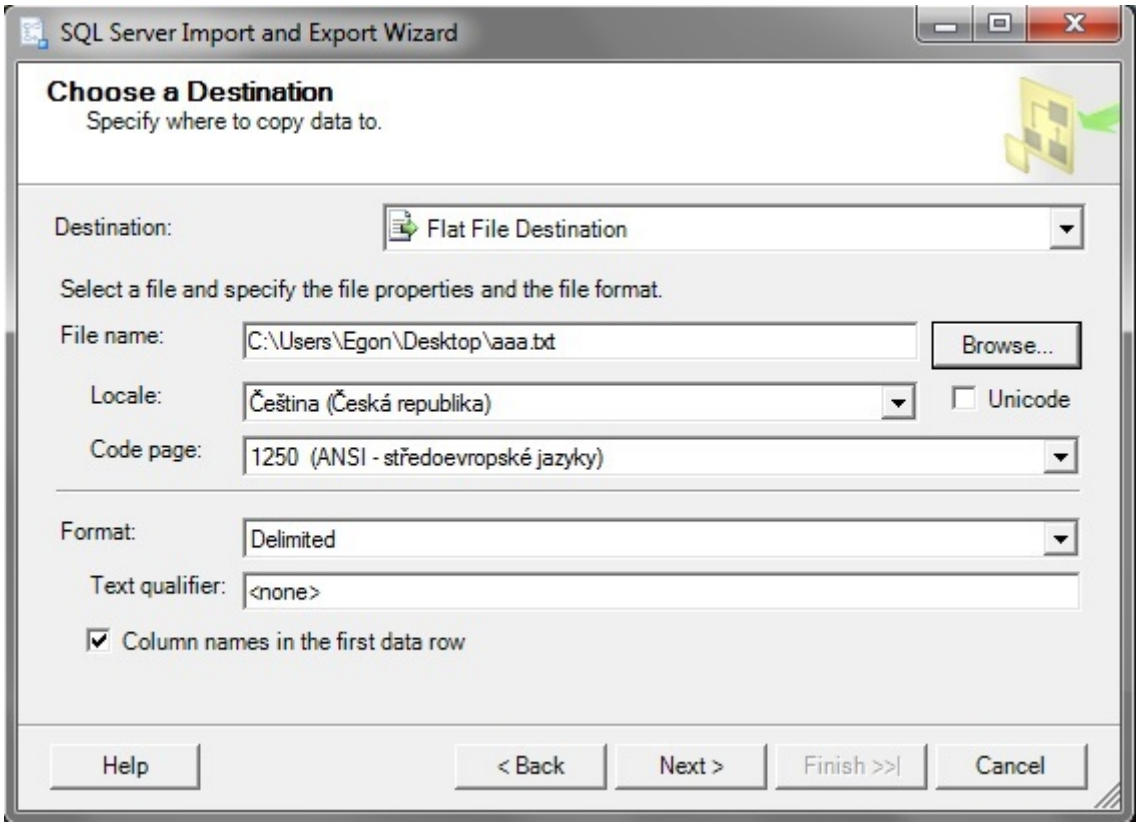
Tento nástroj nabízí nejjednodušší možnost kopírování dat mezi jednotlivými zdroji dat a dále slouží k vytváření jednoduchých databází nebo tabulek. Nástroj umí kopírovat data z nebo do mnoha zdrojů dat, mezi ty základní patří:

- Podnikové databáze - SQL Server, Oracle, a další
- Open source databáze - MySql, PostgreSQL, SQLite, a další
- Soubory Microsoft Office - Microsoft Excel a Microsoft Access
- Prostý databázový soubor (flat file database) – prostá databáze (často jedna tabulka) uložená v textovém souboru ve formě prostého textu
- Cloudové zdroje - Azure Blob Storage

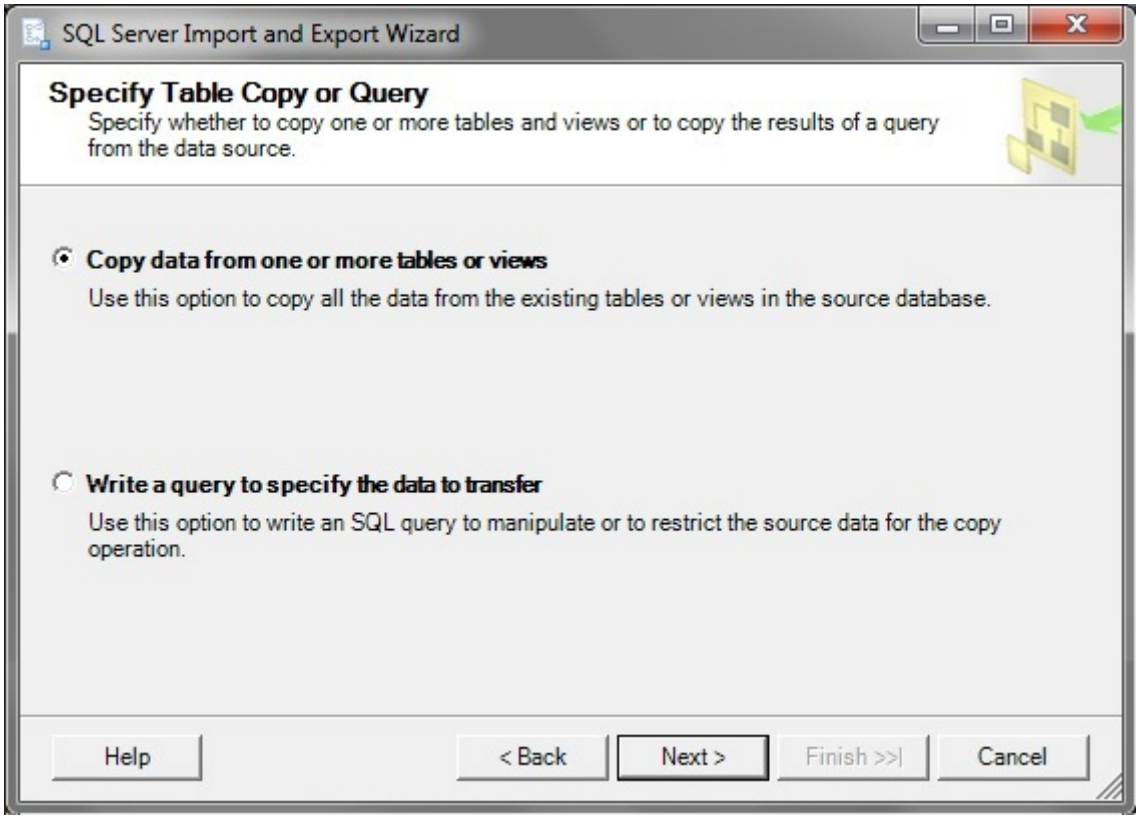
Vlastní kopírování dat se skládá z několika kroků. V prvním kroku se vybere zdroj dat (obrázek 3-7), následně se zvolí destinace, kam se mají vybraná data zkopírovat (obrátek 3-8). Dále je třeba specifikovat, co vše se bude kopírovat. Na výběr jsou 2 varianty (obrázek 3-9). Jednou z nich je výběr zdrojových tabulek, ze kterých se mají data kopírovat. Druhou možností je napsat SQL dotaz.



Obrázek 3-7 - Výběr zdroje dat



Obrázek 3-8 - Výběr destinace pro kopírování dat

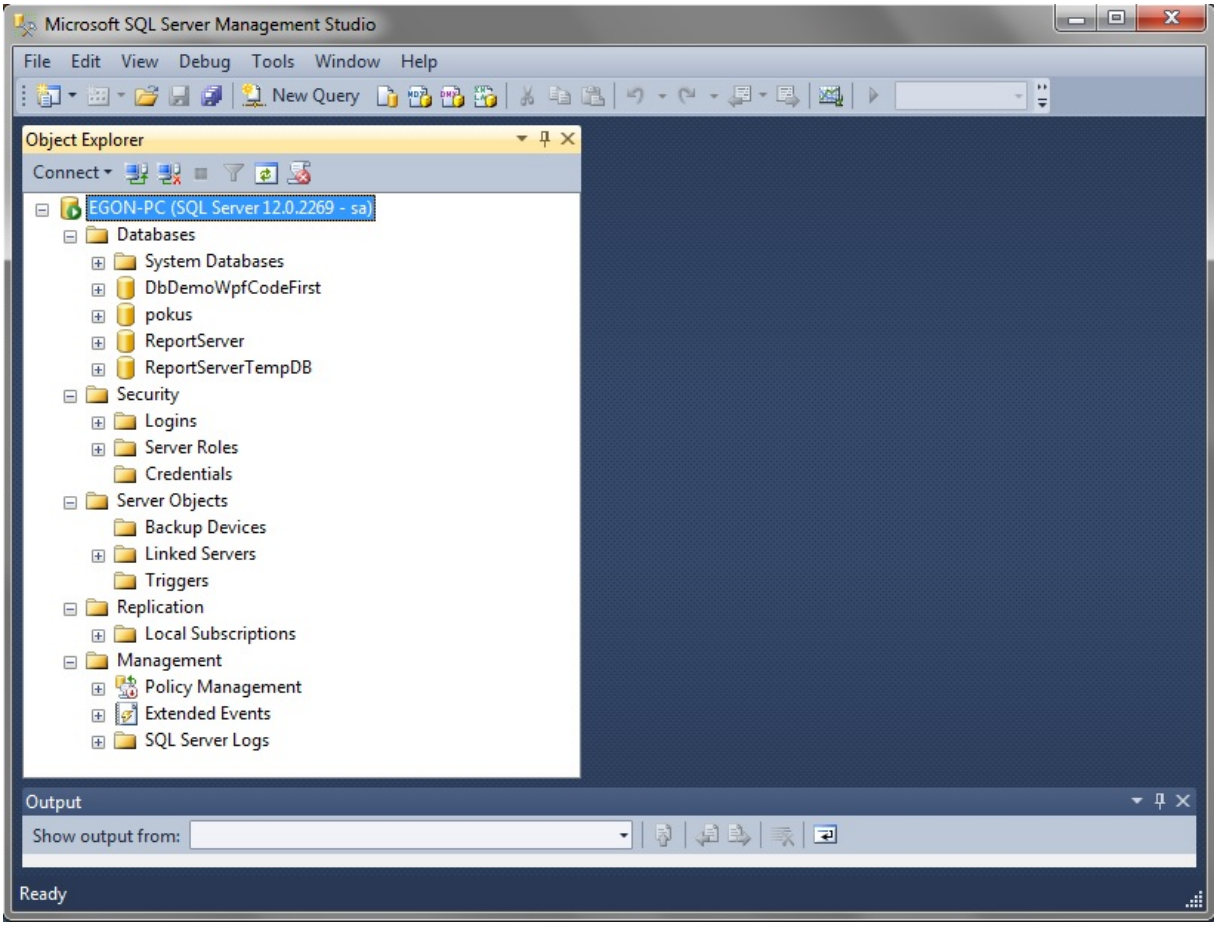


Obrázek 3-9 - Specifikace dat ke kopírování

3.2.4 SQL Management studio

Grafické interaktivní rozhraní tohoto nástroje usnadňuje správu serverů, databází i zdrojů. Tento nástroj umožňuje správu lokálních i vzdálených serverů tak, že se nejprve provede připojení k příslušné instanci SQL Serveru, na které lze následně provádět úlohy související se správou databázového systému. Mezi tyto úlohy patří například procházení a editace jednotlivých databázových objektů, spouštění SQL příkazů a mnoho dalších.

Po spuštění nástroje se zobrazí okno **Object Explorer** (obrázek 3-10), to umožňuje prohlížet a připojovat se k instancím SQL Serveru, ke službám Analysis Services, ke službám Integration Services a ke službám Reporting Services. Po připojení k severu lze procházet všechny jeho komponenty ve stromovém zobrazení.

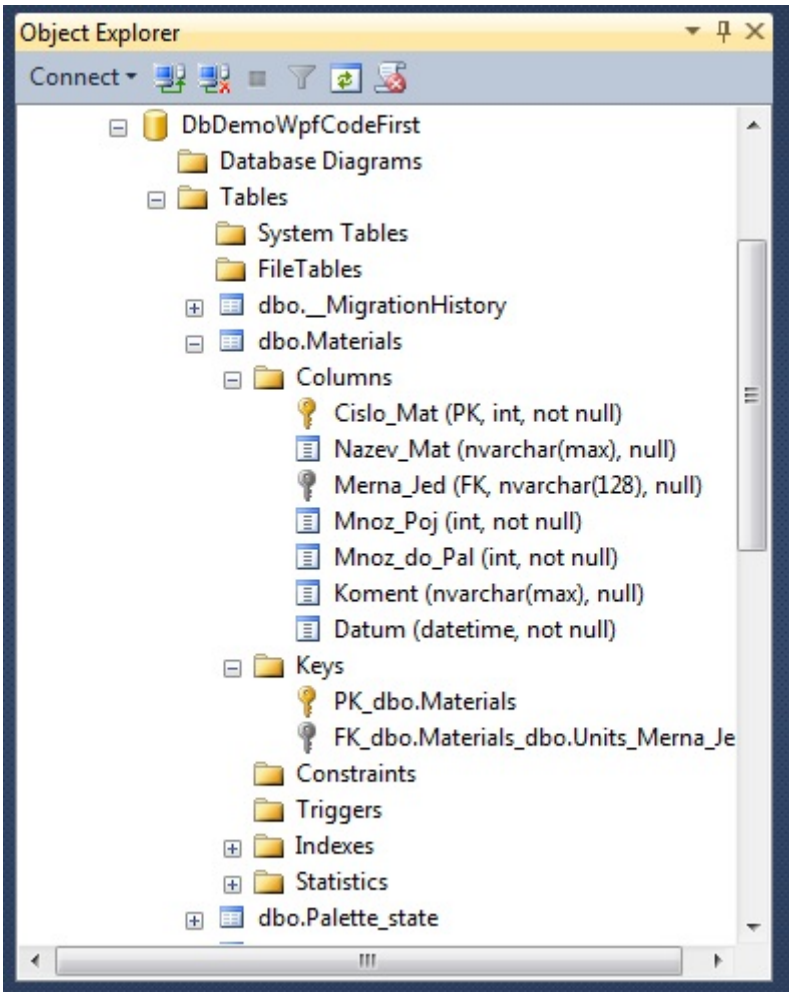


Obrázek 3-10 - SQL Server Management Studio

Po připojení se k instanci serveru bude umožněn přístup k následujícím komponentám a funkcím:

Databases – správa všech systémových a uživatelských databází. U každé databáze jsou následující položky:

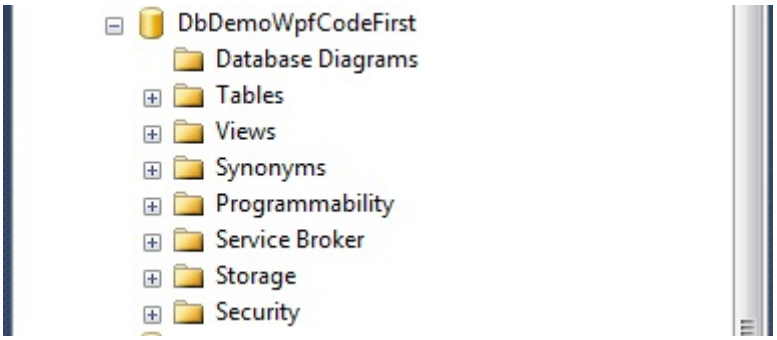
- **Database Diagrams** – nástroj pro vizuální znázornění tabulek a jejich vazeb
- **Tables** – seznam tabulek databáze, každá tabulka má následující položky: Columns (seznam sloupců tabulky), Keys (primární a cizí klíče), Constraints (validace řádků a defaultní hodnoty sloupců), Triggers (seznam všech automaticky spouštěných událostí při modifikaci tabulky), Indexes (seznam indexů), Statistics (seznam statistických záznamů). (Obrázek 3-11)



Obrázek 3-11 - Položky vybrané tabulky

- **Views** – seznam pohledů v této databázi.
- **Synonyms** – seznam synonym v této tabulce.
- **Programmability** – seznam programových rozšíření databáze, obsahuje funkce, procedury, datové typy a odkazy na .NET knihovny.
- **Service Broker** – seznam a funkce technologie pro zajištění bezpečné komunikace s ostatními procesy.
- **Storage** – seznam fulltextových katalogů pro provádění pokročilého vyhledávání.
- **Security** – definuje zabezpečení na úrovni databáze, tedy seznam uživatelů a rolí.

Všechny tyto položky jsou vidět na následujícím obrázku 3-12.



Obrázek 3-12 - Položky u vybrané databáze

Security – seznam zabezpečení serveru. Obsahuje seznam přihlašovacích loginů, rolí databáze a externích účtů. Definuje bezpečnost na úrovni celého serveru.

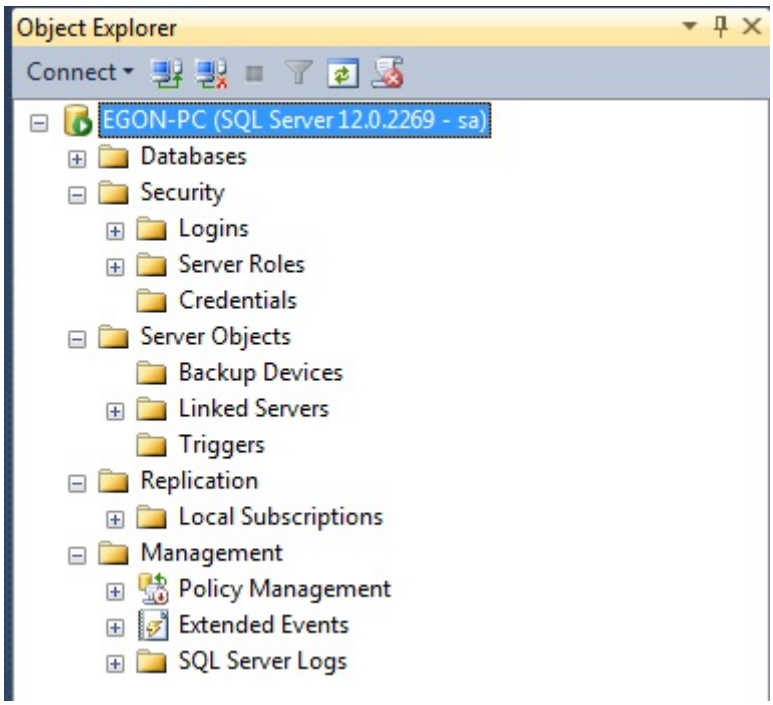
Server Objects – seznam objektů společných pro celý server nebo instanci. Obsahuje položky:

- **Backup devices** – seznam zálohovacích zařízení.
- **Linked servers** – seznam napojení na cizí servery. Dovoluje připojovat databáze z jiné instance.
- **Triggers** – seznam automaticky spouštěných událostí vyvolaných změnami databázového serveru.

Replication – nástroje a seznamy pro replikaci databází na jiný server.

Management – nástroje a funkce pro prohlížení protokolů událostí SQL Serveru, vytváření, prohlížení a správu plánů údržby. Dále jsou zde možnosti konfigurace sběru dat, nástroje pro správu prostředků Resource Governor a další.

Zmíněné položky s jejich funkcemi a nástroji jsou vidět na obrázku 3-13.



Obrázek 3-13 - Další položky Object Exploreru

Tímto bylo řečeno vše podstatné k základnímu popisu a základním možnostem MS SQL Serveru. Další kapitola bude zaměřena na uživatele a možnosti zabezpečení serveru, neboť zabezpečení je realizováno především přes nastavení rolí a práv jednotlivých uživatelů.

4 Uživatelé a zabezpečení

Před samotným založením uživatelů je třeba popsat jednotlivé entity a objekty, se kterými mohou uživatelé přijít do styku. Dále je třeba podívat se na samotné zabezpečení přístupu a práce s databázemi.

4.1 Entity zabezpečení a zabezpečené objekty

Entita zabezpečení – entita, která může využívat prostředky serveru, databáze nebo schématu.

- **Úroveň systému Windows**: doménový účet, lokální účet, skupina
- **Úroveň SQL Serveru**: serverová role, účet v SQL Serveru, účet v SQL Serveru mapovaný na asymetrický klíč, certifikát nebo účet Windows.
- **Úroveň databáze**: uživatel databáze, uživatel databáze mapovaný na asymetrický klíč, certifikát nebo účet Windows, aplikační role, databázová role, databázová role public.

Zabezpečený objekt – objekt, u kterého lze zabezpečit přístup.

- **Úroveň server**: servery/aktuální instance, databáze, koncový bod, účet, serverová role
- **Úroveň databáze**: aplikační role, sestavení, asymetrický klíč, certifikát, kontrakt, databázová role, fulltextový katalog, typ zprávy, vazba na vzdálenou službu, cesta, schéma, služba, symetrický klíč, uživatel
- **Úroveň schéma**: agregát, integritní omezení, funkce, procedura, fronta, statistika, synonymum, tabulka, typ, pohled, kolekce schématu XML

4.2 Oprávnění u zabezpečených objektů:

V této podkapitole jsou definována jednotlivá oprávnění, která mohou být udělena uživatelům:

Alter any – přiřazuje oprávnění vytvářet, měnit nebo odebírat z databáze (serveru) jednotlivé entity zabezpečení.

Alter – přiřazuje oprávnění měnit vlastnosti konkrétní entity zabezpečení, s výjimkou vlastnictví. Jestliže entita zabezpečení dostane oprávnění v určitém rozsahu, může měnit, vytvářet nebo odstraňovat veškeré entity zabezpečení v daném rozsahu.

Backup/Dump – přiřazuje oprávnění pro zálohování (výpis).

Control – přiřazuje možnosti podobné vlastnictví. Entita zabezpečení má pro daný objekt zabezpečení veškerá definovaná oprávnění. Při přiřazení oprávnění Control je potřeba zvážit hierarchii bezpečnostního modelu.

Create – přiřazuje oprávnění vytvářet databázový zabezpečený objekt.

Delete – přiřazuje oprávnění odstranit zabezpečený objekt.

Execute – přiřazuje oprávnění spouštět zabezpečené objekty.

Impersonate – přiřazuje oprávnění impersonovat účet nebo uživatele (ztotožnit účet nebo uživatele s jiným dříve definovaným účtem nebo uživatelem).

Insert - přiřazuje oprávnění vkládat data do entity zabezpečení.

Recieve - přiřazuje oprávnění přijímat zprávy od služby Service Broker.

References - přiřazuje oprávnění odkazovat se na danou entitu zabezpečení.

Restore/Load - přiřazuje oprávnění pro obnovu (načítání).

Select – přiřazuje oprávnění prohlížet data uložené v entitě zabezpečení.

Take ownership – přiřazuje možnost přebírat vlastnictví entity zabezpečení.

Update - přiřazuje oprávnění měnit data uložená v entitě zabezpečení.

View definition - přiřazuje oprávnění prohlížet definici entity zabezpečení prostřednictvím jejích metadat (data poskytující informace o jiných datech).

4.3 Oprávnění k příkazům

V této podkapitole jsou definována jednotlivá oprávnění k provádění příkazů, která lze přiřadit uživatelům.

Create databaze – určuje, zdali přihlášený uživatel smí vytvářet databáze. Uživatel musí být v databázi master nebo musí být členem serverové role sysadmin.

Create defualt – určuje, zdali uživatel smí vytvářet výchozí hodnotu pro sloupec v tabulce.

Create function – určuje, zdali uživatel smí vytvářet v databázi uživatelsky definované funkce.

Create procedure - určuje, zdali uživatel smí vytvářet uložené procedury.

Create rule - určuje, zdali uživatel smí vytvářet pravidla pro sloupce v tabulce.

Create table - určuje, zdali uživatel smí vytvářet tabulky.

Create view - určuje, zdali uživatel smí vytvářet pohledy.

Backup database – určuje, zdali uživatel smí zálohovat databázi.

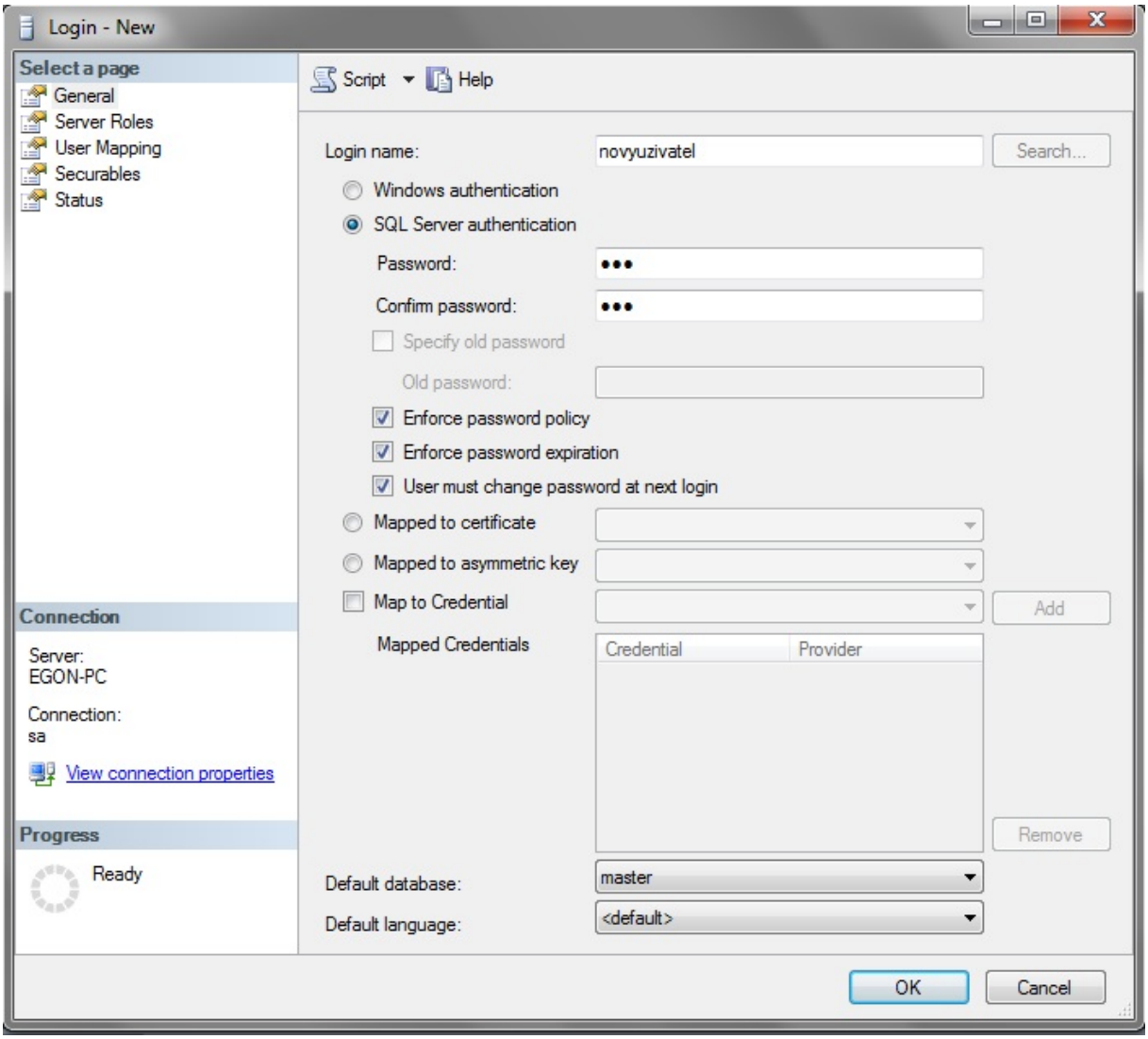
Backup log – určuje, zdali uživatel smí zálohovat transakční protokol.

4.4 Založení nového uživatele a definování jeho rolí

Největší riziko týkající se ztráty nebo poškození u podniků stále hrozí zevnitř. Je větší šance, že o data přijdete chybou zaměstnance spíš než útokem zvenčí podniku. Samotné zabezpečení uvnitř podniku je realizováno pomocí definovaných rolí. U těchto rolí jsou přesně definována vybraná oprávnění, která byla popsána v předchozích podkapitolách. Díky těmto rolím je jasně dané, který uživatel může provádět vybrané operace se serverem nebo databází. Každý uživatel má u svého uživatelského účtu definované role.

4.4.1 Založení nového uživatele

Na obrázku 4-1 jsou vidět možnosti obecného nastavení u nového uživatele, zadává se u něj jméno a heslo, dále se volí způsob autentifikace, volí se standardní databáze a standardní jazyk. Dále se může například zadat možnost vynucení změny hesla po prvním přihlášení nového uživatele. Dále je podstatné u uživatele nastavit výchozí databázi a lze nastavit i výchozí jazyk.



Obrázek 4-1 - Vytvoření nového uživatele

4.4.2 Serverové role

Tyto role slouží k přiřazení oprávnění pro správu serveru. Jestliže se určitý účet stane členem nějaké role, všichni uživatelé, kteří tento účet používají, mohou provádět veškeré úkoly, k nimž má tato role oprávnění. Serverové role jsou následující:

public – výchozí role pro všechny uživatele serveru. Uživatelé dědí oprávnění role public, která obsahuje pouze minimální sadu oprávnění. Jakákoliv jiná role, do níž se uživatel mimo roli public dostane, může jen navýšit jeho oprávnění. Mají-li mít určitá oprávnění všichni uživatelé databáze, je vhodné tato oprávnění přidělit právě roli public.

bulkadmin – umožňuje doménovým účtům provádět dávkové vkládání do databáze. Členové této role mohou do této role přidávat členy a spouštět příkaz BULK INSERT.

Oprávnění: Administer bulk operations

dbcreator – role pro uživatele, kteří potřebují vytvářet, upravovat, mazat a obnovovat databáze. Členové této role mohou do této role přidávat členy a spouštět tyto příkazy: ALTER DATABESE, CREATE DATABASE, DROP DATABASE, EXTEND DATABASE, RESTORE DATABASE, RESTORE LOG.

Oprávnění: Create Database

diskadmin – role pro uživatele, kteří potřebují spravovat soubory na disku. Členové této role mohou do této role přidávat další členy a spouštět uložené procedury sp_addumpdevice a sp_dropdevice.

Oprávnění: Alter resources

processadmin – role pro uživatele, kteří potřebují řídit procesy v SQL Serveru. Členové této role mohou do této role přidávat členy a zastavovat procesy.

Oprávnění: Alter any connection, Alter server state

securityadmin – role pro uživatele, kteří potřebují spravovat přihlašovací jména, vytvářet oprávnění k databázím a číst chybové protokoly. Členové této role mohou přidávat do této role další členy, mohou přidělovat, odebírat a odvolávat oprávnění na úrovni serveru i databáze, resetovat hesla a číst chybové protokoly. Kromě toho mohou spouštět tyto uložené procedury a příkazy: sp_addlinkedsrvlogin, CREATE LOGIN, ALTER LOGIN, DROP LOGIN, sp_droplinkedsrvlogin, GRANT CONNECT, DENY CONNECT, sp_helplogins a sp_remoteoption.

Oprávnění: Alter any login

serveradmin – role pro uživatele, kteří potřebují nastavovat parametry celého serveru a vypínat server. Členové této role mohou do této role přidávat další členy a pouštět následující příkazy a uložené procedury: DBCC FREEPROCCACHE, RECONFIGURE, SHUTDOWN, sp_configure, sp_fulltext_service a sp_tableoption.

Oprávnění: Alter any ednpoint, Alter resources, Alter server state, Alter settings, Shutdown, View server state

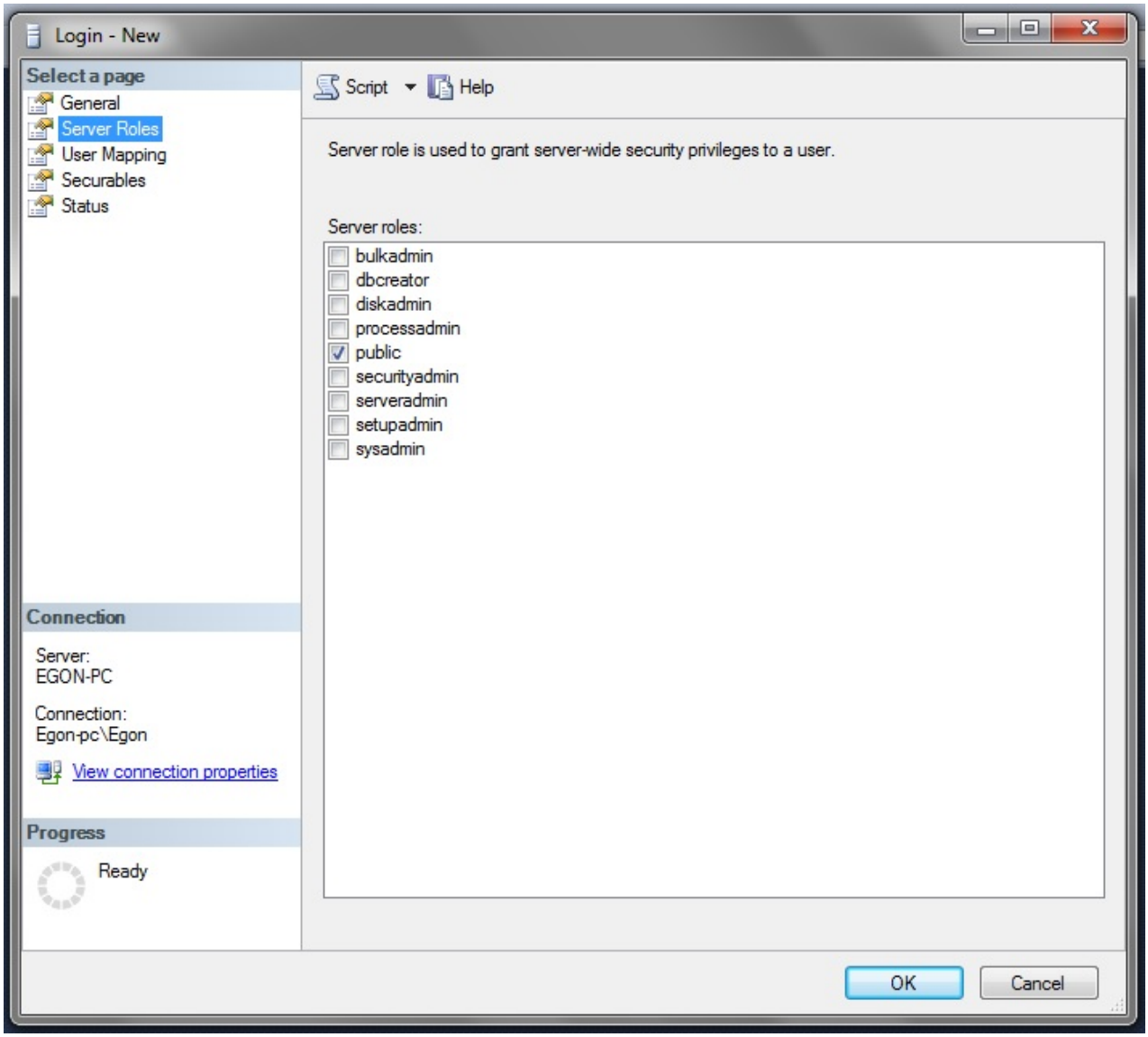
setupadmin – role pro uživatele, kteří potřebují spravovat propojené servery a řídit spouštěcí procedury. Členové této role mohou do této role přidávat další členy, přidávat, mazat a nastavovat propojené servery a řídit spouštěcí procedury.

Oprávnění: Alter any linked server

sysadmin – role pro uživatele, kteří potřebují mít kompletní kontrolu nad SQL Serverem a nainstalovanými databázemi. Členové této role mohou v SQL Serveru provádět jakoukoliv činnost.

Oprávnění: Control Server

Na obrázku 4-2 je vidět přehled serverových rolí, které lze přiřadit uživateli při zakládání jeho účtu.



Obrázek 4-2 - Serverové role

4.4.3 Databázové role

Slouží k přiřazování oprávnění na úrovni databáze. Tyto role se přiřazují pro jednotlivé databáze. Každá databáze má tedy svou množinu rolí. Předdefinované role, které nelze měnit:

public – stejně tak jako u serverové role public, se jedná o roli s minimálně sadou oprávnění. Jakákoliv jiná role, do níž se uživatel mimo roli public dostane, může jen navýšit jeho oprávnění. Mají-li mít určitá oprávnění všichni uživatelé databáze, je vhodné tato oprávnění přidělit právě roli public.

db_accessadmin – role pro uživatele, kteří potřebují přidávat nebo odstraňovat přihlašovací účty k databázi.

Oprávnění: Alter any user, Connect with grant option, Create schema, View any database

db_backupoperator – role pro uživatele, kteří mají zálohovat databázi.

Oprávnění: Backup database, Backup log, Checkpoint, View any database

db_datareader – role pro uživatele, kteří potřebují prohlížet data v databázi. Členové této role mohou číst data ze všech ze všech uživatelských tabulek v databázi.

Oprávnění: Select, View any database

db_datawriter – role pro uživatele, kteří potřebují přidávat nebo upravovat data v uživatelských tabulkách. Členové této role mohou provádět následují operace na všech objektech ve zvolené databázi: DELETE, INSERT a UPDATE.

Oprávnění: Delete, Insert, Update, View any database

db_ddladmin – role pro uživatele, kteří potřebují spouštět úkoly související s definičním datovým jazykem (DDL) v SQL Serveru. Členové této role mohou spouštět všechny příkazy DDL s výjimkou GRANT, REVOKE či DENY.

Oprávnění: Alter any assemble, Alter any Asymetric kay, Alter any certificate, Alter any contract, Alter any database ddl trigger, Alter any database event notification, Alter any dataspace, Alter any fulltext catalog, Alter any message type, Alter any remote control service binding, Alter any route, Alter any schema, Alter any serivce, Alter any symmetric key, Checkpoint, Create procedure, Create queue, Create rule, Create synonym, Create table, Create type, Create view, Create xml schema collection, References, View any database

db_denydatareader – slouží k omezení přístupu k datům v databázi na základě přihlášení. Členové této role nesmějí číst data z uživatelských tabulek v celé databázi.

Oprávnění: odpírá SELECT

db_denydatawriter – slouží k omezení oprávnění pro úpravy v databázi na základě přihlášení. Členové této role nesmějí přidávat, upravovat ani odstraňovat data z uživatelských tabulek v celé databázi.

Oprávnění: odpírá DELETE, INSERT, UPDATE

db_owner: slouží pro uživatele, kteří potřebují mít kompletní kontrolu nad všemi aspekty databáze. Členové této role mohou přiřazovat oprávnění, upravovat nastavení databáze, provádět úkoly údržby databáze i veškeré další práce související se správou databáze, včetně jejího odstanění.

Oprávnění: Control with grant option, View any database

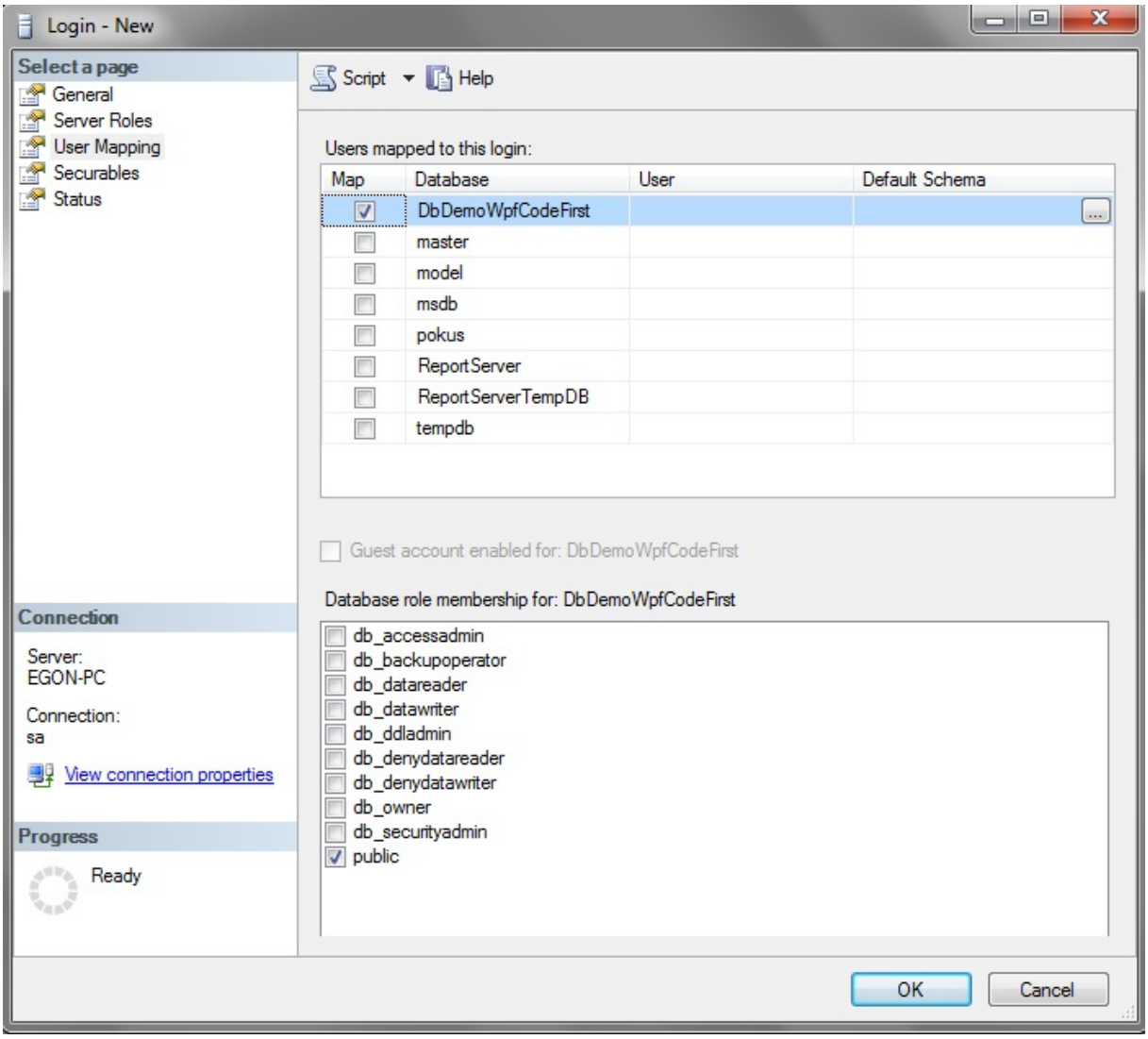
db_securityadmin – role pro uživatele, kteří mají za úkol spravovat oprávnění, vlastnictví objektů a role.

Oprávnění: Alter any application role, Alter any role, Create schema, View definition, View any database

dbm_monitor – role pro uživatele, kteří potřebují sledovat aktuální stav zrcadlení databáze.

Oprávnění: právo View pro zrcadlení databáze, View any diabase

Na obrázku 4-3 je vidět, že se databázové role uživateli přiřazují po označení „User mapping“ v levém panelu. V případě databázových rolí se uživateli tyto role přiřazují pro každou databázi zvlášť.



Obrázek 4-3 - Databázové role

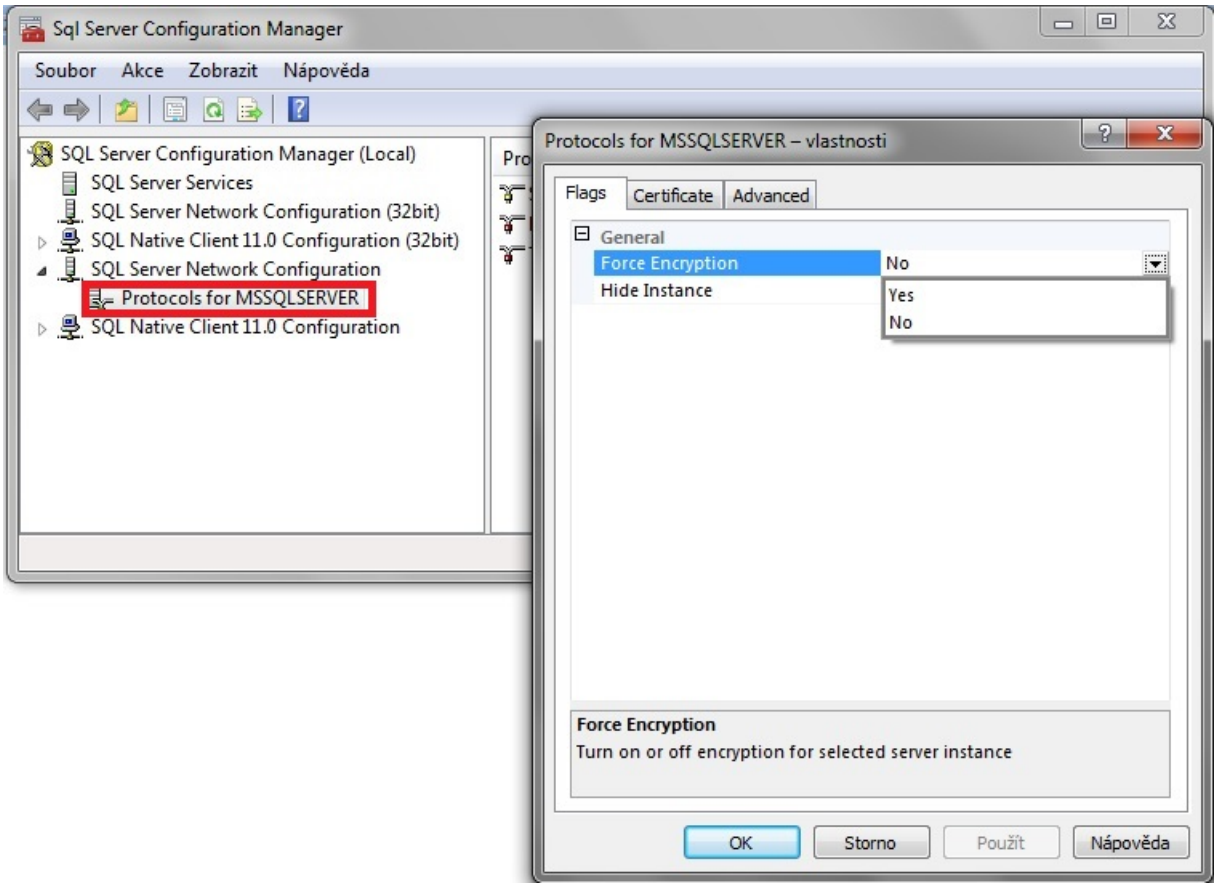
Díky správnému nastavení vlastností a rolí u uživatelských účtů dosáhne toho, že se do vytvořené databáze nedostane nikdo bez řádně vytvořeného účtu. Díky nastavené rolí a z toho plynoucích oprávnění dosáhneme zabezpečení dat vůči poškození nebo ztrátě vinou zaměstnanců. Pro zvýšení zabezpečení vytvořených účtu je možné nastavit například vypršení platnosti přihlášení.

4.5 Šifrování

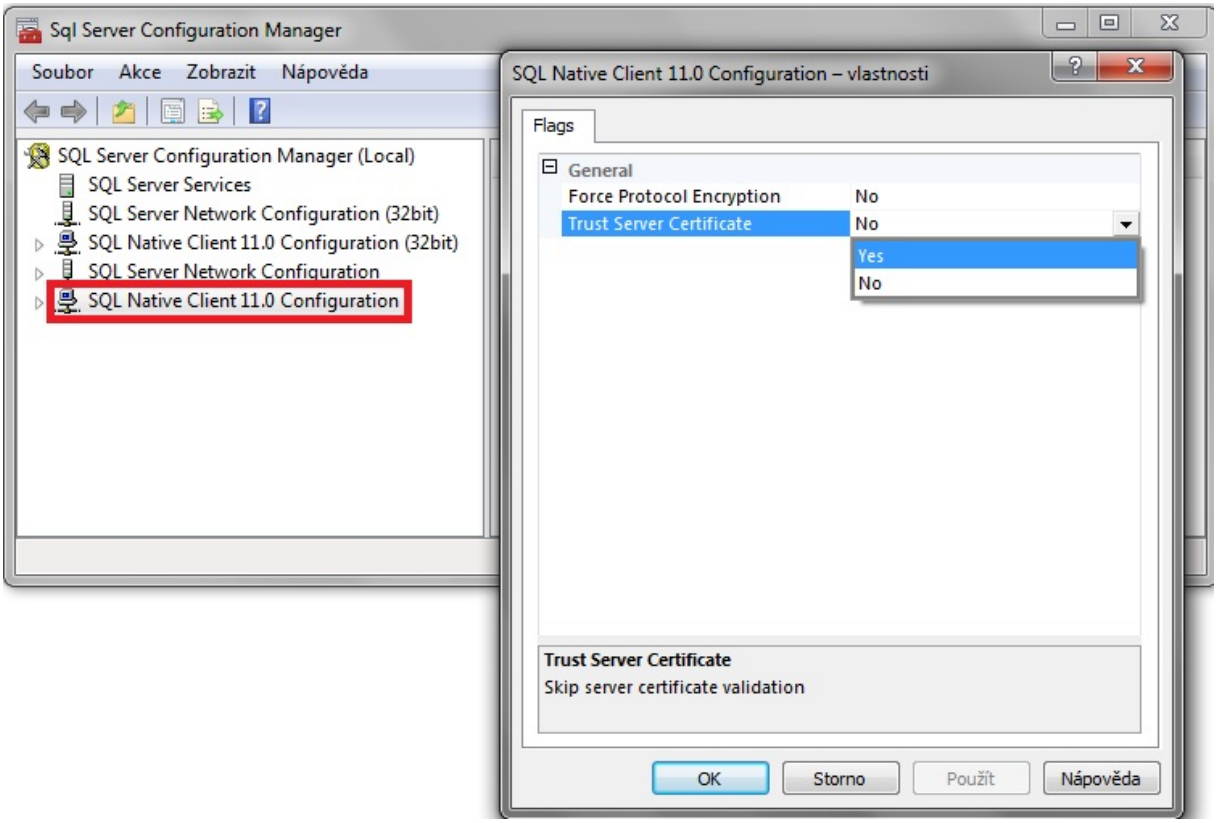
Pokud nestačí uzamčení přístupu k databázím a je vyžadována další úroveň zabezpečení proti neoprávněnému přístupu k datům, využívá se šifrování. Data lze šifrovat buď pro přenos, například pokud je odesíláme webovému serveru, nebo mohou být uložena v šifrovaném formátu. Data jsou šifrována pomocí konceptu šifrovacích klíčů. Tyto klíče mohou být symetrické nebo asymetrické. Dalším konceptem, který je součástí šifrování, jsou certifikáty. Ty jsou v podstatě asymetrickými klíči obsahující zvláštní metadata. Tato metadata obsahují informace, jako je lhůta vypršení platnosti a certifikační úřad, který daný certifikát vydal. [10]

4.5.1 Šifrování přenášených dat

Využívá-li klient rozhraní API SQL Server Native Access Client, všechny požadavky o připojení jsou šifrovány. V připojení nejsou šifrovány pouze pakety přihlášení. Celé šifrování může být volitelně šifrováno po dobu trvání tohoto připojení. Požadavek šifrovaného kanálu může být vynucen serverem (aby byla všechna připojení implicitně šifrována) nebo může šifrování požadovat klient vytvářející připojení. Pro vynucení šifrování u serveru je třeba v nástroji SQL Server Configuration Manager z místní nabídky zvolit Protocols for MSSQLSERVER a ve vlastnostech změnit položku ForceEncryption na Yes. To je ukázáno na obrázku 4-4. V záložce Certificate lze následně vybrat certifikát, který je již nainstalovaný v místním systému, aby jej SQL Server použil k šifrování dat. Pokud certifikát nevyberete a přesto zvolíte vynucení šifrování, SQL Server použije k šifrování dat svůj certifikát, který podepsaný sám sebou. Aby tento certifikát používat klienti, musí se ve vlastnostech položky SQL Native Client Configuration změnit položka Trust Server Certificate na Yes. To je ukázáno na obrázku 4-5. Dále je zde možnost změnit položku Force Protocol Encryption na Yes, díky které bude po klientovy vždy vyžadováno, aby vytvářel šifrované připojení. [10]



Obrázek 4-4 - Změna položky ForceEncryption



Obrázek 4-5 - Změna položky Trust Server Certificate

4.5.2 Šifrování statických dat

Rozsah citlivých informací v databázích se neustále rozšiřuje. U těchto informací je nutné kromě zabezpečení pomocí oprávnění přidat další vrstva ochrany. Toho lze dosáhnout pomocí symetrických klíčů, které vytváří SQL Server. Při prvním spuštění SQL Serveru se vytvoří zvláštní symetrický klíč pro server, zvaný Service Master Key (SMK). Tento klíč se používá k šifrování všech klíčů databáze (Database Master Key, DMK) a všech tajných údajů na úrovni serveru, jako jsou údaje o pověření nebo přihlašovací hesla k propojenému serveru. Při šifrování dat se klíčem SMK dešifruje klíč DMK, aby klíč DMK mohl pro klienta dešifrovat data. Pro každou databázi existuje pouze jeden klíč DMK. Klíč DMK se používá pouze k šifrování dat, nevytváří se tedy implicitně. [10]

K tomuto vytvoření slouží příkaz ve tvaru:

USE <<databáze>>
CREATE MASTER KEY
ENCRYPTION BY PASSWORD='<<heslo>>'

5 Zálohování

Databáze jsou místem, kde se nacházejí veškeré informace, se kterými se v podniku pracuje (adresáře, informace o zákaznících, znalosti), aby byla firemní data v bezpečí a aby byla data vždy k dispozici, je potřeba mít solidní plán zálohování a obnovy databází. Díky zálohování se lze ochránit před ztrátou dat, narušením databází nebo selháním softwaru.

5.1 Plán pro zálohování a obnovu

Vytvoření a nasazení plánu pro zálohování a obnovu dat je jednou z nejdůležitějších povinností administrátora. K náhodnému mazání nebo narušení dat dochází neustále, proto je třeba na tyto plány brát zřetel.

Pro vytvoření tohoto plánu je třeba si položit tyto otázky:

Jaký typ databází se má zálohovat? Pro různé databáze platí různé požadavky. Databáze master nezbytná pro všechny operace MS SQL Serveru (při jejím selhání selže celý server) nepotřebuje být zálohována každou hodinu, stačí záloha po vytvoření nové databáze nebo po změně konfiguračních parametrů. Naproti tomu uživatelskou databázi zpracovávající transakce v reálném čase je třeba zálohovat častěji.

Jak významná jsou data v určité databázi? To může být klíčem k určení jak a kdy dané databáze zálohovat. Například u vývojové databáze postačí plná záloha jednou týdně. U databáze s objednávkami, která se během dne mění, je třeba dělat plné zálohy častěji a doplnit je navíc o každodenní rozdílové zálohy a o hodinové zálohy transakčních protokolů.

Jak často dochází v databázi ke změnám? Toto je další určující parametr pro rozhodnutí, jak často databázi zálohovat. Například databázi určenou pouze pro čtení, která se tedy nemění, není třeba zálohovat pravidelně. Databázi, která se vždy mění v určitou dobu, je třeba zálohovat po provedení těchto změn. Databázi, která se mění pravidelně, je třeba zálohovat průběžně.

Jak rychlá musí být obnova dat? Dalším určujícím parametrem je rychlost obnovy ztracených dat. U kritických databází je třeba dostat databázi do provozního stavu velmi rychle. U těchto databází je třeba nastavit čas pro obnovu.

Je k dispozici vybavení pro zálohování? Pro zálohování je třeba mít příslušný hardware. Pokud tento hardware není k dispozici, není možno zálohovat. Pravidelné zálohování vyžaduje několik zálohovacích zařízení a několik sad zálohovacích médií.

Lze zálohy komprimovat? Vyšší verze SQL Serveru podporují komprimované zálohy a jakákoliv verze umí tyto zálohy obnovit. Komprimovaná záloha je menší, ačkoliv obsahuje stejná data, proto zabírá méně prostoru v uložišti a zároveň se tím zvyšuje rychlost zálohování i obnovy. Tato výhoda je kompenzována vyšší zátěží procesoru, tím lze negativně ovlivnit celkový výkon serveru. Je-li komprimovaná samotná databáze, není třeba komprimovat zálohy, protože samotné zmenšení záloh není tak výrazné.

Na jaký čas v průběhu dne je nejlepší naplánovat zálohování? Zálohování by mělo probíhat v čase, kdy je databáze využívána nejméně. Tím se zrychlí proces zálohování.

Je potřeba ukládat zálohy mimo firmu? V případě přírodní katastrofy je pro obnovu systému nutné ukládat kopie záloh mimo prostory firmy. U těchto záloh je třeba i kopie softwaru nutného pro obnovu práce.

5.2 Základní typy záloh

Základní typy záloh jsou následující:

Plná záloha databáze – zahrnuje všechny objekty, systémové tabulky i data. Po spuštění zálohování zkopíruje SQL Server veškerý obsah databáze a zároveň zahrne část transakčních protokolů potřebných pro obnovu dat z doby během zálohování. Plnou zálohu lze použít ke kompletní obnově stavu dat v databázi v momentu dokončení zálohování.

Rozdílová záloha – slouží jako záloha dat, která se změnila od poslední plné zálohy. Jelikož se ukládají pouze rozdíly, trvá vytvoření této zálohy kratší dobu a je možno ji provádět častěji. Stejně jako u plné zálohy, i rozdílová záloha obsahuje části transakčních protokolů potřebných pro obnovu databáze k momentu dokončení zálohy.

Záloha transakčních protokolů – tyto protokoly představují sériový záznam všech změn v databázi a používají se během operací obnovy k dokončení hotových transakcí. Při tomto zálohování se do zálohy ukládají změny, k nimž došlo od poslední zálohy transakčního protokolu. Následně se protokol zkrátí, díky tomu se smažou transakce, které byly dokončeny nebo zrušeny. Na rozdíl od předchozích dvou zmíněných typů zálohování se do záloh transakčního protokolu ukládá stav tohoto protokolu ve chvíli spuštění zálohování, nikoliv ve chvíli, kdy zálohování skončí.

Záloha souborů a skupin souborů – tato záloha umožňuje zálohovat soubory nebo skupiny souborů, nikoliv celou databázi. To může být užitečné v případě, kdy se jedná o obrovskou databázi a je třeba ušetřit čas. I v tomto případě je nutné zálohovat transakční protokol. Pokud objekty v databázi zasahují do více souborů či skupin souborů, je nutné zálohovat všechny dotčené soubory či skupiny souborů.

5.3 Modely obnovení

SQL Server používá následující modely obnovení:

Simple – jednoduchý model obnovení slouží pro databáze, u kterých je nutná obnova do doby poslední zálohy. V tomto případě by se zálohovací strategie měla skládat z plných a rozdílových záloh. U tohoto modelu obnovení nelze zálohovat transakční protokoly, SQL Server v tom případě pomocí služby Truncate Log on Checkpoint maže v kontrolním bodě neaktivní záznamy z transakčního protokolu. Tento model je ideální pro většinu systémových databází.

Full – plný model obnovení je určen pro databáze, které je nutno obnovit až do dobu selhání či do konkrétního okamžiku. U tohoto modelu se všechny operace zaznamenají do protokolu, včetně dávkových operací a dávkového načítání dat. Zálohovací strategie tohoto modelu by měla zahrnovat plné a rozdílové zálohy, ale i zálohy transakčních protokolů nebo pouze plné plné zálohy a zálohy transakčních protokolů.

Bulk-logged – dávkový model obnovení snižuje velikost transakčních protokolů, i přesto si uchovává většinu flexibility plného modelu obnovení. U tohoto modelu se dávkové operace a dávkoné načítání zapisují do protokolu pouze minimálně a není možné mít pod kontrolou každou jednotlivou operaci. Pokud databáze selže před plnou nebo rozdílovou zálohou, je nutné, aby byly dávkové operace a dávkové načítání provedeny znovu. U tohoto modelu by měla zálohovací strategie zahrnovat plné a rozdílového zálohy i zálohy transakčních protokolů nebo pouze plné zálohy a transakčních protokolů.

Každá databáze může mít rozdílný model obnovení.

5.4 Výběr zálohovacího zařízení a média

Tento výběr ovlivňuje velká spousta faktorů, například:

- **Kapacita** – množství dat, která je v rutinním provozu nutno zálohovat.
- **Spolehlivost** – spolehlivost hardwaru a médií je určující pro to, jak užitečné budou vytvořené zálohy v případě potřeby obnovení ztracených dat.
- **Rozšiřitelnost** – rozšiřitelnost zálohovacího řešení předurčuje možnost zvýšení jeho původní kapacity.
- **Rychlost** – je třeba zvážit rychlost, kterou se budou data zálohovat a následně obnovovat.
- **Cena** – výsledné volba je často ovlivněna i cenou zálohovacího zařízení.

Možná zálohovací zařízení a média:

Páskové jednotky – nejčastější zálohovací média. Tyto jednotky používají magnetické kazety a na ně ukládají data. Tyto pásky jsou relativně levné, nicméně nejsou příliš spolehlivé. Páska se může přetrhnout nebo protáhnout. Navíc mohou pásky v průběhu času ztrácet informace. Průměrná kapacita páskových jednotek se pohybuje ve stovkách gigabajtů. Ve srovnání s ostatními jsou magnetické pásky pomalé.

Jednotky Digital Audio Tape (DAT) – tyto jednotky pomalu nahrazují standardní páskové jednotky z pozice preferovaného typu. Těchto jednotek existuje mnoho formátů, nejpoužívanější formáty jsou Digital Linear Tape (DLT) a Super DLT. Kapacita těchto pásek se pohybuje ve stovkách gigabajtů. U větších firem lze zvážit pásky Linear Tape Open (LTO) s vyšší kapacitou (LTO-5 až 1500 GB).

Páskové systémy s automatickou výměnou – neboli páskové knihovny využívají řadu pásek, díky čemuž vzniká kapacita schopná dostát vysokým požadavkům. Systém automatické výměny umožňuje automatickou změnu pásky podle potřeb zálohování a obnovy. Většina těchto systémů používá pásky DAT s formátem DLT, SDLT nebo LTO. Jednotky DLT jsou schopny nahrát až 45 GB za hodinu, rychlost lze zvýšit pořízením systému s více jednotkami, potom lze nahrávat na více pásek zároveň. Většina jednotek SDLT a LTO je schopna nahrát až stovky gigabajtů dat za hodinu. **Příklad**: Systém s šestnácti jednotkami LTO, dosahovaná rychlost více než 13,8 TB za hodinu. Do systému se vejde až 500 pásek s celkovou kapacitou více než 800 TB.

Vyjímatelné diskové jednotky – vyjímatelné disky, například externí jednotky USB nebo disky eSATA, se čím dál častěji používají i pro účely zálohování. Tyto jednotky nabízejí slušnou rychlost a jsou snadno použitelné pro zálohu jediné jednotky nebo systému. Nevýhodou můžu být vyšší cena.

Diskové jednotky – představují nejrychlejší způsob zálohování a obnovy dat. Oproti páskám se jedná o zrychlení z řádů hodin na řády minut. V případě nutné rychlé obnovy se pevným diskům žádné médium nevyrovná. Oproti předchozím je nevýhodou vysoká cena.

Diskové zálohovací systémy – tyto systému představují kompletní řešení pro zálohování a obnovu, využívají velká pole disků, díky čemuž dosahují vysokých výkonů. Vysoké spolehlivosti lze docílit v případě diskového pole RAID, které přináší rezervu pro toleranci k chybovosti. Typické systémy této kategorie využívají technologii virtuální knihovny, díky čemuž je systém Windows vnímá jako páskové systémy s automatickou výměnou. Výsledkem je snadnější práce s těmito systémy. **Příklad**: Systém se 128 virtuálními jednotkami a 16 virtuálními knihovnami na uzel. Celková kapacita se pohybuje kolem 7,5 TB na uzel. Pří kompletním využití má toto řešení kapacitu až 640 TB a lze přenést 17,2 TB za hodinu.

Výběr zálohovacího zařízení (systému) představuje zásadní krok při tvorbě plánu zálohování a obnovy.

5.5 Zálohovací strategie

Z typu databáze a typu dat vycházejí jednotlivé zálohovací strategie.

Strategie pro uživatelské databáze:

Obnova v řádu minut – podle možností provádět plné zálohy dvakrát týdně. Užívat noční rozdílové zálohy a transakční protokol zálohovat během pracovní doby každých 10 minut. Neměla by se použít možnost Truncate Log On Checkpoint, neboť při jejím zapnutí nebude možné obnovit určité transakce. Pro zrychlení zálohování/obnovy je třeba používat více zálohovacích zařízení.

Obnova k určitému pracovnímu bodu - podle možností provádět plné zálohy dvakrát týdně. Užívat noční rozdílové zálohy a transakční protokol zálohovat během pracovní doby každých 10 minut. Neměla by se použít možnost Truncate Log On Checkpoint. Je třeba používat pojmenované transakce a vkládat do transakčního protokolu pojmenované značky. Pro zrychlení zálohování/obnovy je třeba používat více zálohovacích zařízení.

Obnova v řádu hodin - podle možností provádět plné zálohy dvakrát týdně. Užívat noční rozdílové zálohy, transakční protokol je během pracovní doby třeba ukládat každých 60 minut. Nepoužívat možnost Truncate Log On Checkpoint. Pro zrychlení zálohování/obnovy je třeba používat více zálohovacích zařízení.

Obnova změn v daném dni - plné zálohy provádět nejméně jednou týdně. Užívat noční rozdílové zálohy, transakční protokol je během pracovní doby třeba ukládat podle potřeby na základě velikosti protokolu a pauzy mezi zálohami. Nepoužívat možnost Truncate Log On Checkpoint.

Read-only – naplánovat pravidelnou plnou zálohu na každých 30-60 dní a doplnit ji dostatečnou plnou zálohou vždy, když dojde k nějaké úpravě.

Strategie pro systémové databáze:

distribution – musí být dostupná v momentě nastavování replikace a ve chvíli, kdy server hraje roli distributora. Zálohy by se měly naplánovat po vytvoření snímků. U transakční replikace se neplánuje pravidelné zálohování protokolu.

master – plné zálohování se pustí vždy po vytvoření či odstranění určité databáze, při změně velikosti databáze, přidání nebo odstranění účtu či uprávě konfigurace serveru. U této databáze je podstatné uchovávat několik sad záloh.

model – s touto databází se pracuje stejně jako s databází read-only.

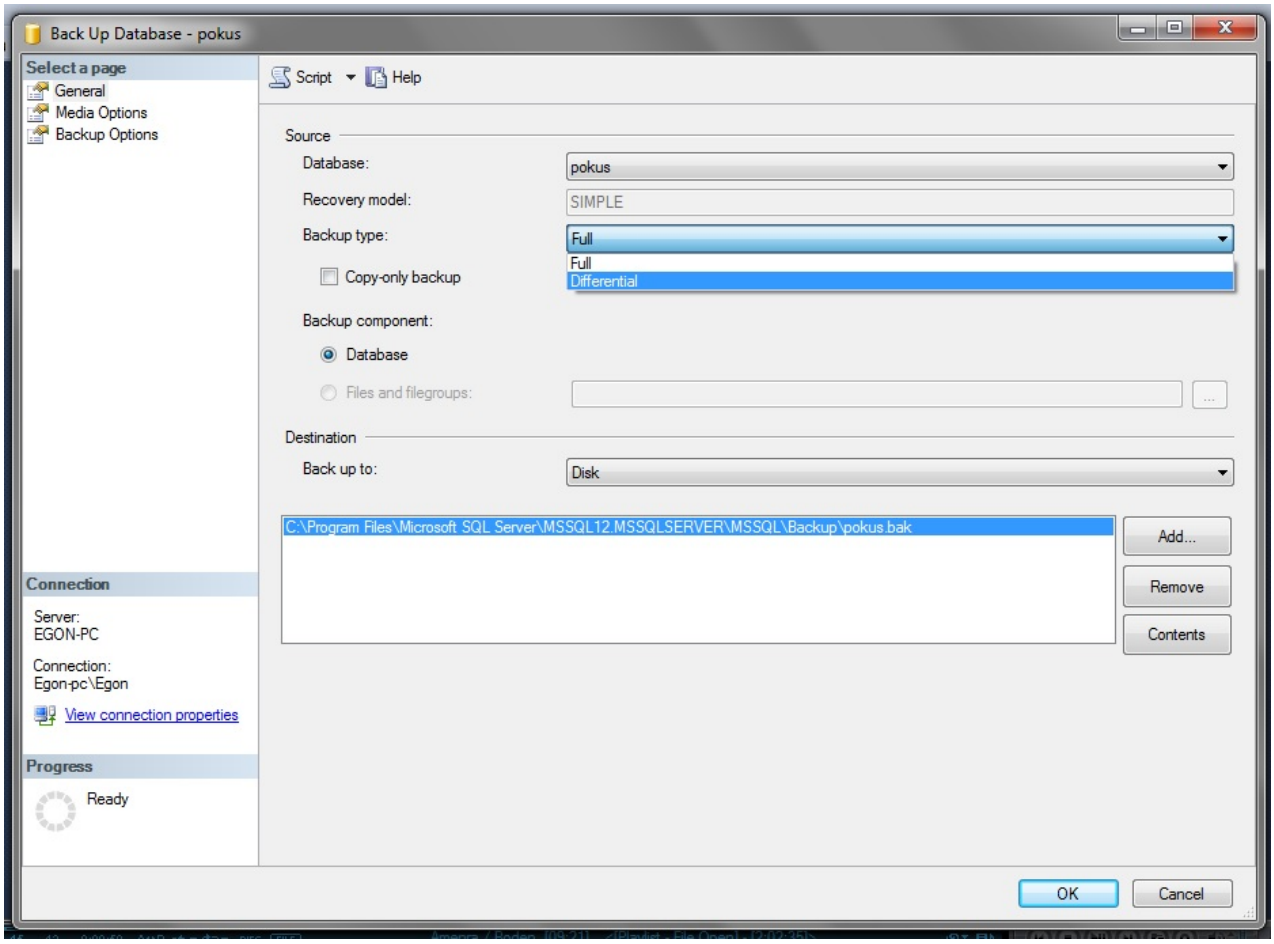
msdb – pokud se plánují úkoly pomocí služby SQL Server Agent, zálohuje se tato databáze pravidelně, protože právě v této databázi se uchovávají plány a historie provádění úkolů, dále je v ní uložena historie zálohování.

publication - musí být dostupná v momentě nastavování replikace a ve chvíli, kdy server hraje roli vydavatele. Pokud se nezálohují protokoly, je třeba tuto databázi zálohovat vždy, když se změní nastavení replikace.

subscription - musí být dostupná v momentě nastavování replikace a ve chvíli, kdy server hraje roli odběratele. Tato databáze by neměla být starší než nejkratší retenční interval ze všech publikací, k nimž je odběratel přihlášen.

tempdb – běžně není nutno tuto databázi zálohovat. Tato databáze se znovu vytvoří vždy, když je spuštěn SQL Server.

Je třeba pamatovat na to, že při každém spuštění SQL Severu se obnoví všechny databáze, odvolají se nepotvrzené transakce a dokončí se potvrzené transakce, jejichž změny nebyly v době zastavení instance ještě zapsány na disk. Na obrázku 5-1 jsou vidět možnosti nastavení zálohování. Vybírá se databáze, kterou chceme zálohovat, typ zálohy a zálohovací zařízení.



Obrázek 5-1 - Možnosti zálohování

6 Rozšířené služby

V této kapitole jsou popsány další služby systému MS SQL Server, které nesouvisí s údržbou nebo správou serveru, ale mohou ve strojírenských podnicích najít využití.

6.1 SQL Server Analysis Services

Tato služba přináší online analytické zpracování (OLAP) a data mining funkcionalitu pro aplikace business intelligence. Služba vám dovoluje navrhovat, vytvářet a spravovat multidimenzionální struktury, které obsahují data agregovaná z jiných zdrojů, jako jsou třeba relační databáze.

Pro data mining aplikace je zde možnost navrhnout, vytvořit a vizualizovat data mining modely, které jsou sestaveny z dalších datových zdrojů díky využití různých průmyslových data mining algoritmů.

V analysis services jsou podstatné především tyto objekty:

Zdroje dat – ty jsou u Analytic Services reprezentovány především connection stringem, který definuje jak se Analytic Services připojí k fyzickým datům. Je zde definováno jméno serveru, databáze, zabezpečení a další informace týkající se připojení. Analytic Services podporují mnoho zdrojů dat. Mezi ně patří MS SQL Server databáze, ale i databáze vytvořené v jiných softwarech jako třeba Oracle, DB2 nebo Teradata.

Pohledy zdrojů dat – ty obsahují logický model schématu používaného databázovými modely. Jmenovitě jsou to kostky, dimenze a mining struktury. Pohled zdroje dat je tvořen metadaty ve formátu XML Pohled zdroje dat obsahuje metadata, která reprezentují vybrané objekty z jednoho nebo více základních zdrojů dat, nebo metadata, která budou použita pro generování základních relačních zdrojů dat. Mohou být vytvořen pro jeden nebo více zdrojů dat zároveň, tím pádem se můžou zvolit pro OLAP nebo data mining objekty u různých zdrojů. Mohou obsahovat vazby, klíče, jména objektů, počítaná pole a dotazy, které nejsou prezentovány v základních datech nebo které existují odděleně od základních zdrojů dat.

Kostky – jsou sadou souvisejících veličin a dimenzí, které jsou používány pro analýzu dat. Dané veličiny jsou faktické a transakční, jejich hodnoty mohou být tím, k čemu se chce uživatel přiblížit. Zdrojem veličin jsou sloupce v jedné nebo více tabulkách a jsou seskupené do skupin velečin. Dimenze je skupina atributů, které reprezentují oblast zájmu spojenou s veličinami v kostce a používají se pro analýzu veličin v kostce. Kostka je rozšířena o kalkulace, indikátory klíčových výkonů, akce, perspektivy a překlady. Kostka je v podstatě synonymem pro Unified Dimensional Model (UDM). V SQL Serveru je kostka založena na tabulkách a pohledech, které jsou namodelovány vůči zdroji dat. Kostka může být vyvinuta i bez základního relačního zdroje dat.

Dimenze – jsou stěžejními komponentami kostek. Dimenze organizují data s vazbami k oblasti zájmu, jako jsou zákazníci, dodavatelé nebo zaměstnanci. Dimenze obsahují atributy, které korespondují se sloupci v dimenzionálních tabulkách. Tyto atributy vystupují jako atributy hierarchií. Mohou být organizovávány do uživatelem definovaných hierarchií nebo mohou být definovány jako dědičné hierarchie založené na sloupcích v základních dimenzionálních tabulkách. Hierarchie se používají k organizování veličin, které jsou obsaženy v kostce.

Data Mining Struktury – jsou datové struktury, které definují datovou doménu, ze které jsou sestavy dolovacích modelů. Jedna taková struktura může obsahovat více dolovacích modelů, které sdílejí stejnou doménu. Stavebními kostkami dolovacích struktur jsou sloupce

dolovacích struktur, které popisují, jaká data obsahuje datový zdroj. Tyto sloupce obsahují informace jako datový typ, obsahový typ a jak jsou data distribuována. Dolovací struktura zároveň obsahuje zahnízněné tabulky. Ty reprezentují vazby 1:N mezi entitami a jejich atributy.

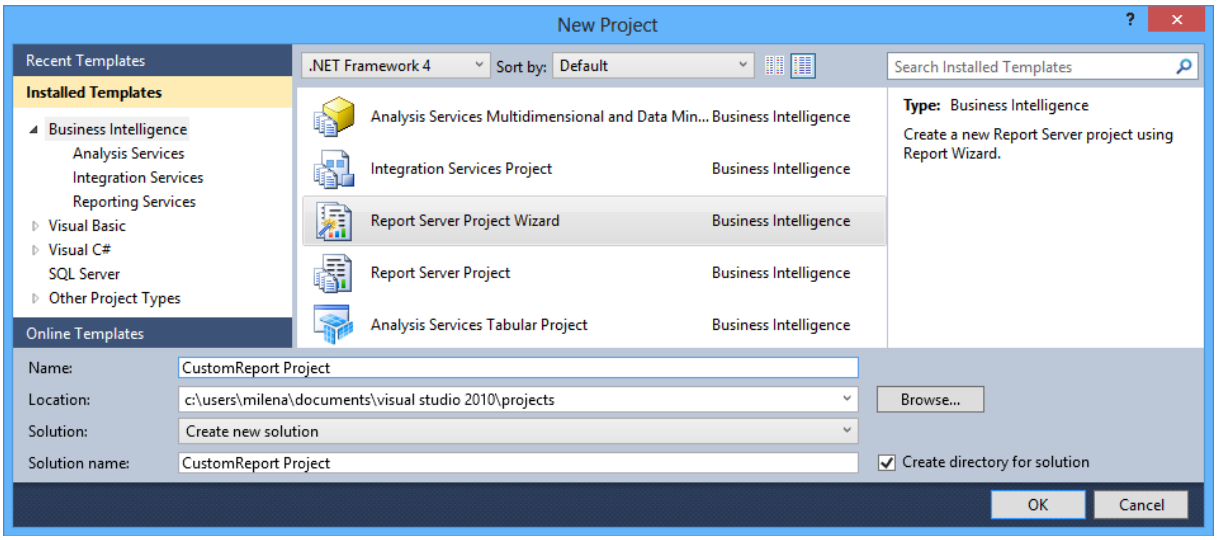
Data Mining Model – aplikuje dolovací algoritmus na data, která jsou reprezentována dolovací strukturou. Model také obsahuje sloupce, je obsažen v dolovací struktuře a dědí všechny hodnoty vlastností, které jsou definovány strukturou. Obsahuje dvě vlastnosti a to algoritmus a využití sloupců.

Deployment Wizard je služba, která zajišťuje distribuci dat získaných pomocí analytických služeb.

6.2 SQL Server Reporting Services

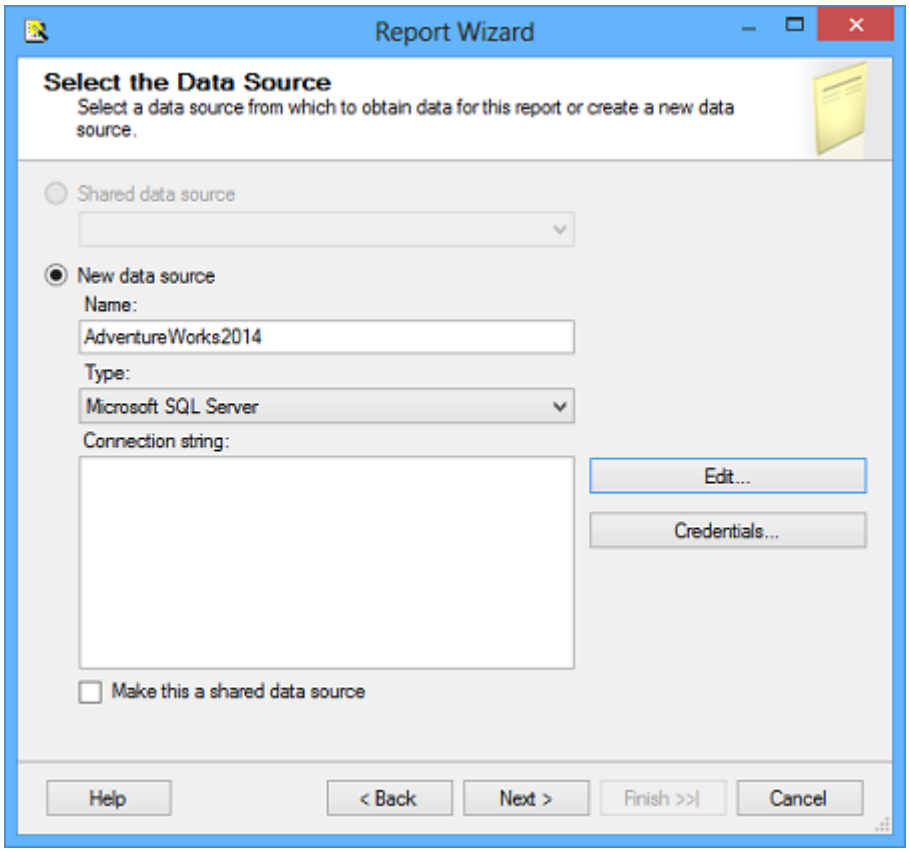
Jedná se o službu, která umožňuje vytvářet a spravovat širokou škálu různých typů reportů a ty pak převádět do různých formátů. Lze vytvořit základní report, který obsahuje pouze tabulky a diagramy, nebo komplexnější vizualizace dat s grafy, mapami a sparkliny. Reporty lze tvořit z dat ze serverové databáze ale také z dalších relačních databází jako je Oracle a dalších typů multidimenzionálních datových zdrojů. Postup tvorby reportu je následující:

V aplikaci SQL Data Tools založíme nový projekt. Pod záložkou Business Inteligence vybereme Report server project wizard. (obrázek 6-1)



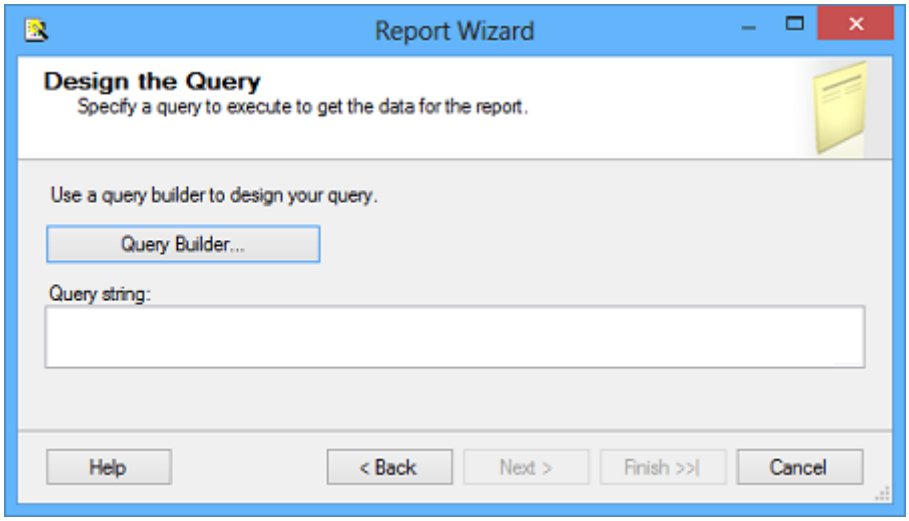
Obrázek 6-1 - Založení projektu [14]

Následně vybereme nový datový zdroj, ze kterého chceme report vytvořit, typ nastavíme Microsoft SQL Server. (obrázek 6-2)

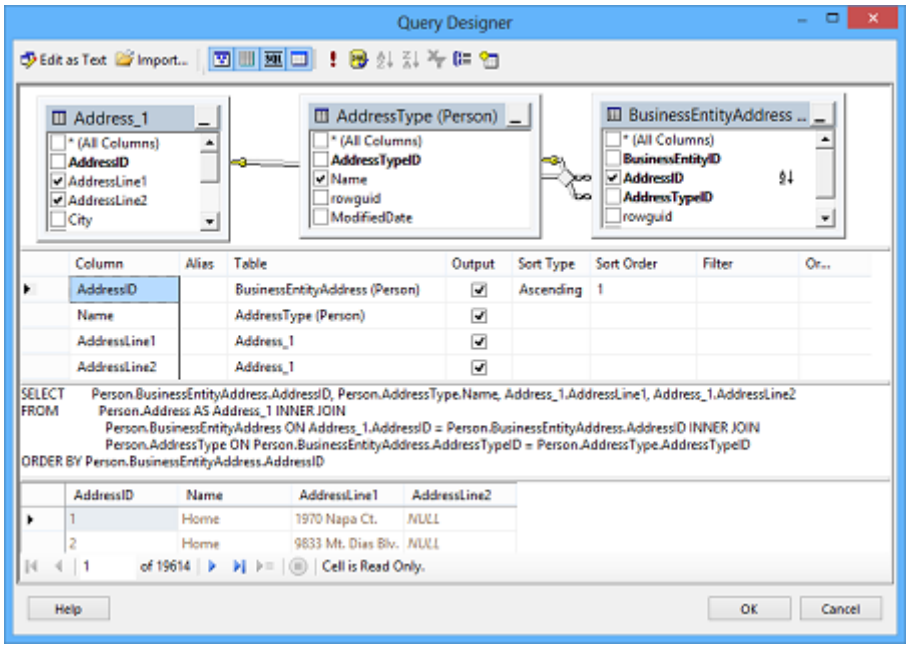


Obrázek 6-2 - Výběr zdroje dat [14]

V dalším kroku se nastaví připojení k databázi. Po otestování připojení se přejde ke specifikaci dotazů. (obrázek 6-3) Zde je možnost přejít na tvůrce dotazu (obrázek 6-4) nebo lze dotaz zapsat ručně do připraveného pole.

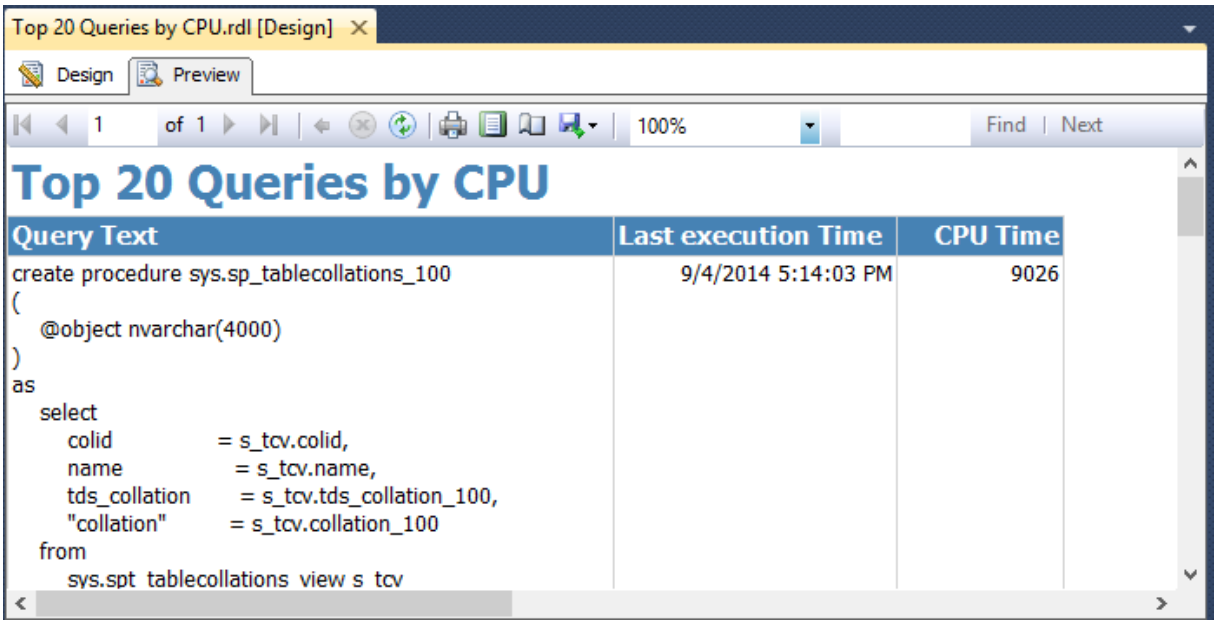


Obrázek 6-3 - Specifikace dotazu [14]



Obrázek 6-4 - Tvůrce dotazu [14]

Na obrázku 6-5 je vidět výsledný report.



Obrázek 6-5 - Výsledný report

7 Ukázka implementace MS SQL Serveru pro vytvořenou aplikaci

V této kapitole je popsán postup vytvoření cvičné aplikace pomocí softwaru Microsoft Visual Studio 2015 s využitím platformy Windows Presentation Foundation (WPF) s nástavbou Entity Framework. Následně je pro databázi této aplikace implementován MS SQL Server. Jsou zde popsány jednotlivé kroky nastavení serveru zmíněné v předchozích kapitolách jako například vytvoření uživatelů a nastavení zálohování.

7.1 Windows Presentation Foundation a Entity Framework

V této podkapitole je popsána zvolená platforma pro vytvoření cvičné aplikace spolu s nástavbou Entity Framework.

Nejdříve je však nutné definovat softwarovou platformu .NET. Tato platforma je souborem technologií v softwarových produktech. Jedná se o run-time prostředí pro tvorbu a běh aplikací, znamená to, že jsou aplikace pro něj napsané s tímto prostředím pevně spojeny. Dále toto prostředí obsahuje širokou řady knihoven a tříd. [11]

7.1.1 Windows Presentation Foundation

WPF je platforma určena pro vývoj a běh formulářových aplikací. Je podmnožinou .NET Frameworku od verze 3.0. Skládá se ze dvou hlavních částí. Jednou z nich je soubor dynamicky linkovaných knihoven (DLL) potřebných pro tvorbu aplikací, které jsou charakterizovány bohatě vizuálním uživatelským prostředím a zároveň silnou a rozsáhlou vazbou na data. Druhou hlavní části je veřejné rozhraní pro tvorbu aplikací (application programming interface, API), které umožňuje vytvořeným aplikacím přístup k výše zmíněným knihovnám. [11]

Tato platforma vznikla částečně kvůli stále rozšířenějšímu vytváření a používání aplikací na chytrých telefonech, pro které nebyly aplikace vytvářené na jiných platformách vhodné.

Základem WPF je komponentová architektura. Platforma nabízí spoustu hotových komponent, ze kterých poskládáme formulář. Mezi tyto komponenty patří například tlačítka, pole, postníky, popisky a další. Dále je zde možnost vytvoření vlastní komponenty. [11]

Tato platforma využívá programovací jazyk **XAML** (eXtensible Application Markup Language). Jedná se o poměrně jednoduchý a všestranně použitelný deklarativní programovací jazyk. Je vhodný pro konstrukci a inicializaci .NET objektů. Tento jazyk vychází z jazyku **XML** (eXtensible Markup Language). XML je značkovací jazyk navržený tak, aby si do něj mohl uživatel přidat své vlastní značky a používat ho k čemukoliv. XAML je tedy XML jazyk uzpůsobený pro tvorbu aplikací.

XAML dokument se skládá z komponent. Struktura těchto komponent je stromová, což znamená, že komponenty v sobě mohou obsahovat libovolné množství dalších komponent. Každý XAML dokument obsahuje právě jednu kořenovou komponentu (většinou formulářové okno), ve kterém jsou umístěny další komponenty.

Příklad deklarace komponenty (tlačítka) pomocí jazyku XAML:

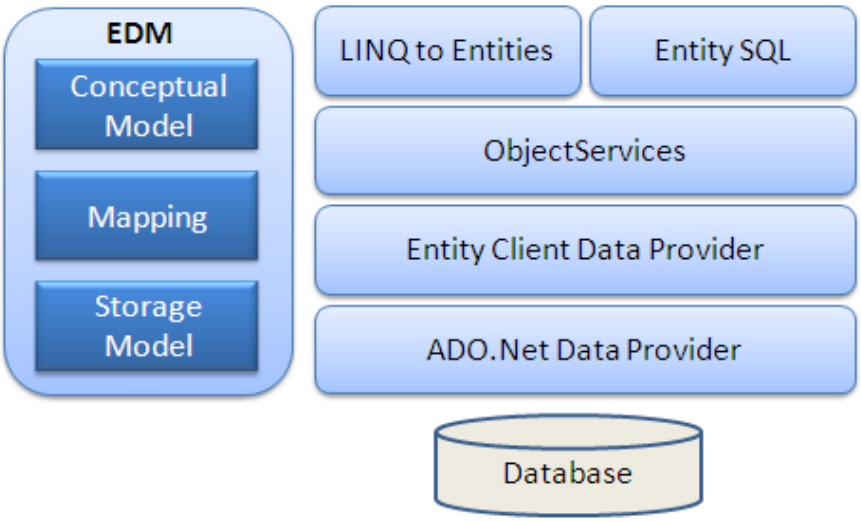
<Button Content="OK" Height ="50" Width="50" />

Tento kód vytvoří tlačítko o velikosti 50 na 50 pixelů a bude obsahovat text „OK".

7.1.2 Entity Framework

Jedná se o objektově-relační mapovací nástavbu pro platformu WPF. Tato nástavba umožňuje tvůrcům aplikací pracovat s relačními daty jako s doménově-specifikovanými objekty. Odpadá tedy potřeba psaní většiny programovacího kódy potřebného pro specifikování přístupu k datům. Tvůrci aplikací tedy mají automatizovaný mechanismus pro přístup k datům a pro jejich uložení v databázi. Implementace této nástavby dovoluje používat služby jako změna stopování, rozlišení identit nebo překlad dotazů. [13]

Na obrázku 7-1 je vidět architektura nástavby Entity Framework. Jednotlivé složky této architektury jsou dále popsány.



Obrázek 7-1 - Architektura Entity Frameworku [13]

EDM (Entity Data Model) se skládá ze tří hlavních částí - Konceptuálního modelu, mapování a skladovacího modelu. [13]

Konceptuální model obsahuje modelové třídy a jejich vazby. Je nezávislý na návrhu tabulek databáze. [13]

Skladovací model je databázový návrhový model, který obsahuje tabulky, pohledy, uložené procedury i s jejich vazbami a klíče. [13]

Mapování je tvořeno informacemi o tom, jak je konceptuální model mapovaný vůči skladovacímu modelu. [13]

LINQ to Entities (L2E) je dotazovací jazyk užívaný k psaní dotazů vůči objektovým modelům. Výsledkem jsou entity, které jsou definovány v konceptuálním modelu. [13]

Entity SQL je další dotazovací jazyk. O něco těžší než jazyk (L2E). [13]

Objektový servis je stěžejní části pro získávání dat z databáze a k jejich opětovnému vrácení. Tento servis je zodpovědný za materializaci, což je proces převodu dat od Entity Client Data Provideru na entitní objektovou strukturu. [13]

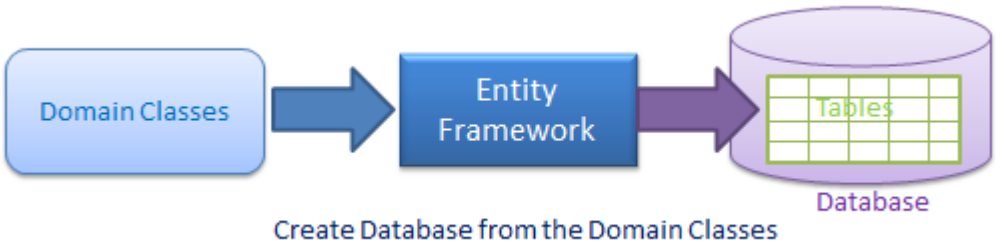
Entity Client Data Provider je zodpovědný za převod dotazů L2E nebo Entity SQL na SQL dotaz, který můžu vykonat základní databáze. Komunikuje s ADO.Net Data Providerem, který pracuje s data v databázi. [13]

ADO.Net Data Provider komunikuje s databází a používá pro to standardní jazyk ADO.Net. [13]

Entity framework podporuje tři různé přístupy vytvoření databáze pro aplikaci:

Code First

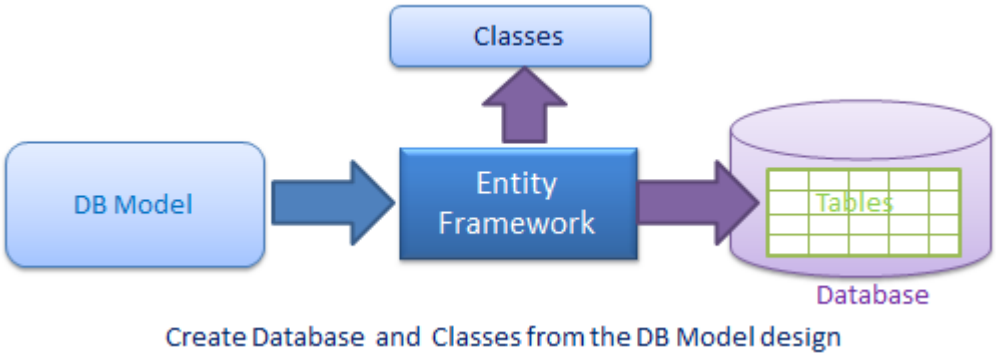
U tohoto přístupu není použit designér pro tvorbu databáze. Místo toho se pomocí kódu vytvoří třídy a z těchto tříd se následně generuje databáze. Princip je vidět na obrázku 7-2.



Obrázek 7-2 - Přístup Code First [13]

Model First

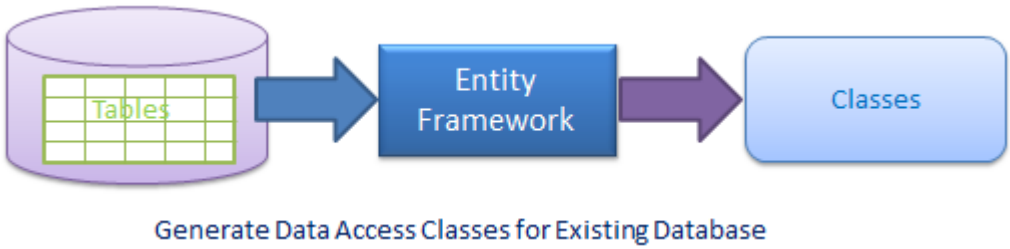
U tohoto přístupu se pomocí designéru vytvoří model, který je tvořen entitami, vazbami a hierarchií dědičnosti. Z tohoto modelu se následně vytvoří jednotlivé třídy a zároveň databáze. Princip je vidět na obrázku 7-3.



Obrázek 7-3 - Přístup Model First [13]

Database First

U tohoto přístupu se z již existující databáze vygenerují třídy. U tohoto přístupu se používá designér stejně jako u přístupu Model First, na rozdíl od předchozího přístupu se v designéru zvolí již existující databáze a z této databáze se vyberou tabulky, pohledy a procedury, pro které se mají vytvořit třídy.



Obrázek 7-4 - Přístup Database First [13]

Ve své práci jsem si zvolil přístup Code First. Vytvořil jsem tedy nejdříve třídy pomocí kódu, zároveň jsem v kódu definoval primární a cizí klíče. Následně jsem z těchto tříd vygeneroval databázi. Postup je vidět v dalších kapitolách.

7.2 Datová analýza

V této podkapitole je provedena datová analýza ukázkové aplikace. Jsou popsány jednotlivé entity a vztahy mezi nimi. Jedná se o aplikaci pro evidenci materiálů a palet. Kromě těchto dvou hlavních entit se v aplikaci vyskytují tři číselníky. Hodnoty těchto číselníků budou výchozími hodnotami pro vybrané atributy hlavních entit.

7.2.1 Entitní typy

Hlavní entitní typy:

Materiál (Material) – tabulka s přehledem všech materiálů s následujícími atributy:

- Číslo materiálu typ: číslo – primární klíč, jednoznačný identifikátor materiálu
- Název materiálu typ: text
- Měrná jednotka typ: text – cizí klíč
- Pojistné množství typ: číslo
- Množství do palety typ: číslo
- Komentář typ: text
- Datum typ: čas

Paleta (Palette) – tabulka s přehledem všech palet s následujícími atributy

- Číslo palety typ: číslo – primární klíč, jednoznačný identifikátor palety
- Typ palety typ: text – cizí klíč
- Stav palety typ: text – cizí klíč
- Adresa uložení typ: text
- Číslo materiálu typ: číslo – cizí klíč
- Množství v paletě typ: číslo
- Komentář typ: text

Stav palety (Palette_state) – tabulka s jednotlivými stavy palet, z těchto hodnot se následně volí hodnota v entitě paleta. Tato entita má následující atributy:

- Stav palety typ: text - primární klíč, jednoznačný identifikátor stavu palety
- Popis typ: text

Typ palety (Palette_type) – tabulka s jednotlivými typy palet, z těchto hodnot se následně volí hodnota v entitě paleta. Tato entita má následující atributy:

- Typ palety typ: text - primární klíč, jednoznačný identifikátor typu palety
- Popis typ: text

Měrná jednotka (Unit) – tabulka s jednotlivými měrnými jednotkami, z těchto hodnot se následní volí hodnota v entitě materiál. Tato entita má následující parametry:

- Měrná jednotka typ: text - primární klíč, jednoznačný identifikátor jednotky
- Popis typ: text

7.2.2 Integritní omezení

- Na jedné paletě je uložen právě jeden materiál
- Stejný druh materiálu může být uložen na více paletách
- Materiál má právě jednu měrnou jednotku
- Stejná měrná jednotka může být u více materiálů
- Paleta je právě jednoho typu

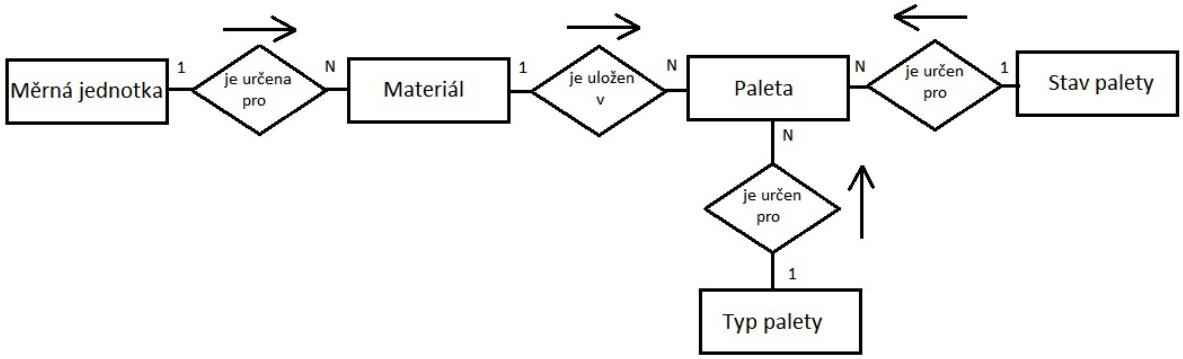
- Stejného typu může být více palet
- Paleta je současně právě v jediném stavu
- Ve stejném stavu může být více palet najednou

7.2.3 ER-diagram

Z výše zmíněných integritních omezení vyplívají následující vztahy

- Materiál - paleta 1:N
- Měrná jednotka – materiál 1:N
- Typ palety – paleta 1:N
- Stav palety – paleta 1:N

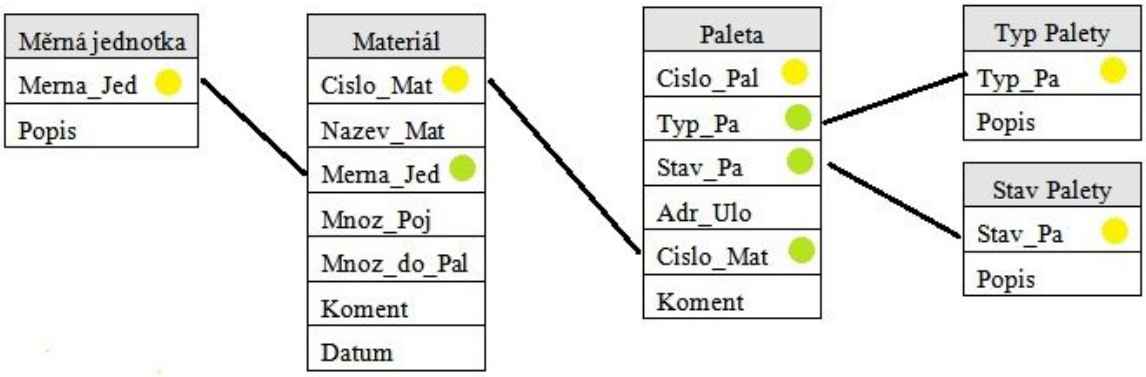
Výsledný ER-diagram je vidět na obrázku 7-5.



Obrázek 7-5 - ER-diagram

7.2.4 Propojení entit pomocí klíčů

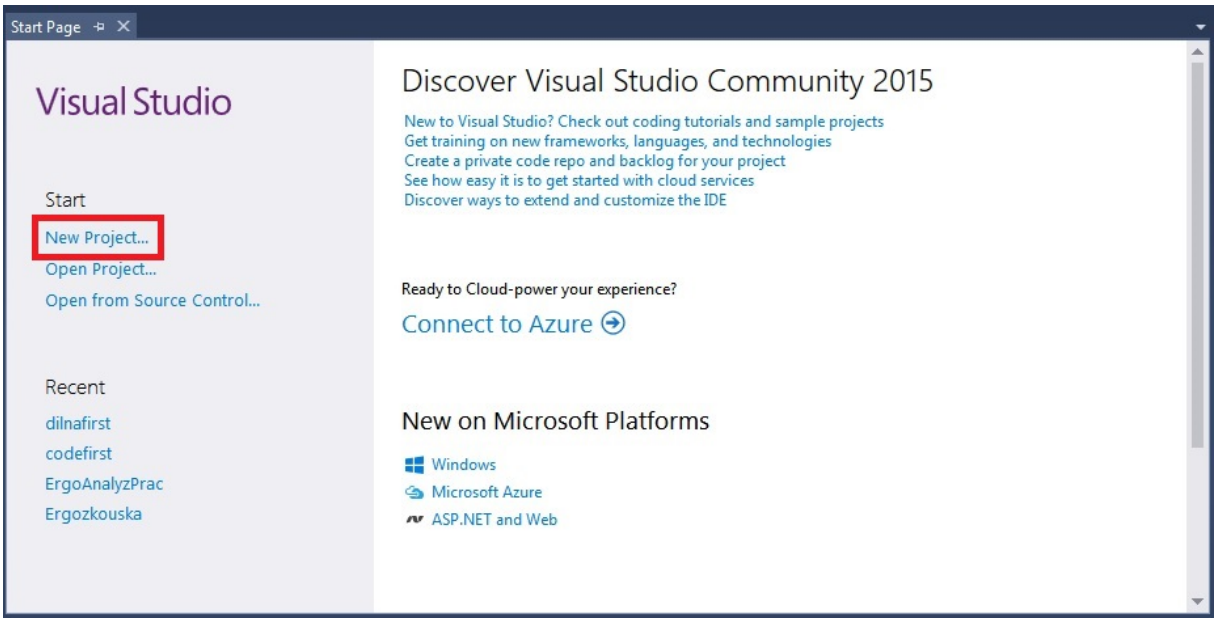
Každá z entit má svůj primární klíč, dále jsou entity mezi sebou provázány pomocí cizích klíčů, to je vidět na obrázku 7-6. Žlutě jsou označeny primární klíče, zelenou barvou klíče cizí.



Obrázek 7-6 - Propojení tabulek pomocí klíčů

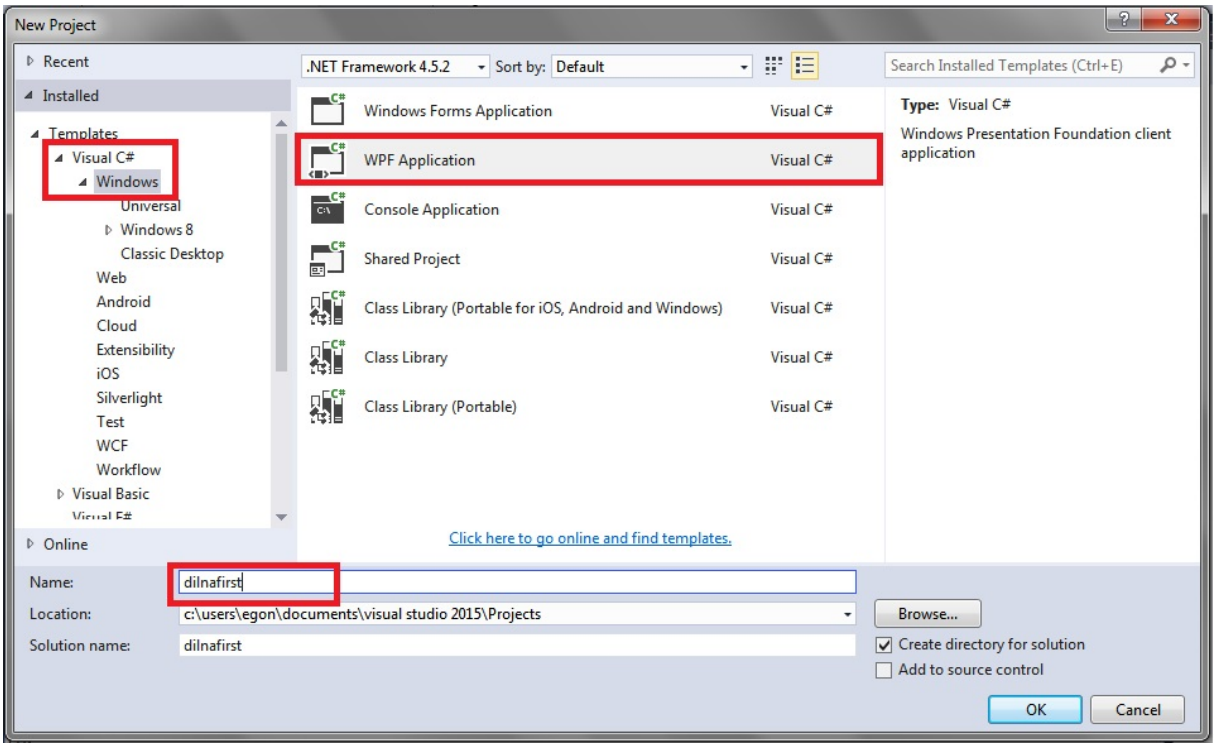
7.3 Vlastní tvorba aplikace

V této podkapitole je krok po kroku popsána tvorba aplikace pomocí platformy WPF. Po spuštění Visual Studia 2015 se zvolí možnost založit nový projekt, jak je vidět na obrázku 7-7.



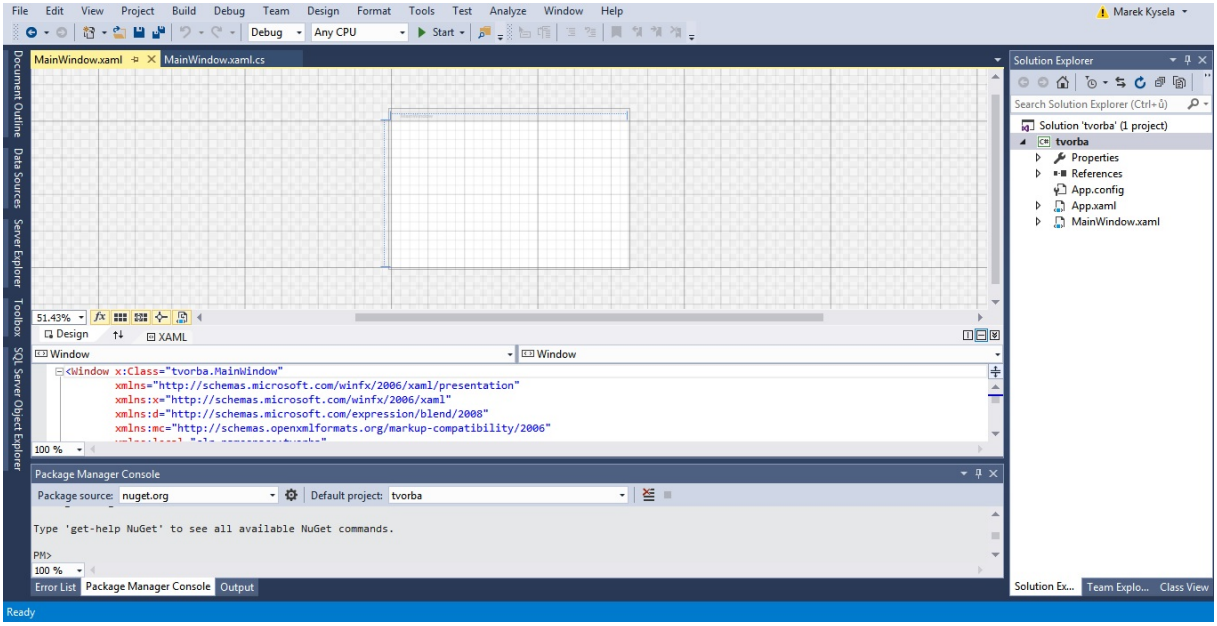
Obrázek 7-7 – Založení nového projektu

Po otevření podokna je třeba zvolit v levém sloupci Templates – Visual C# a následně z menu vybrat WPF Application. Poté se v dolní části podokna zadá název projektu. (obrázek 7-8)



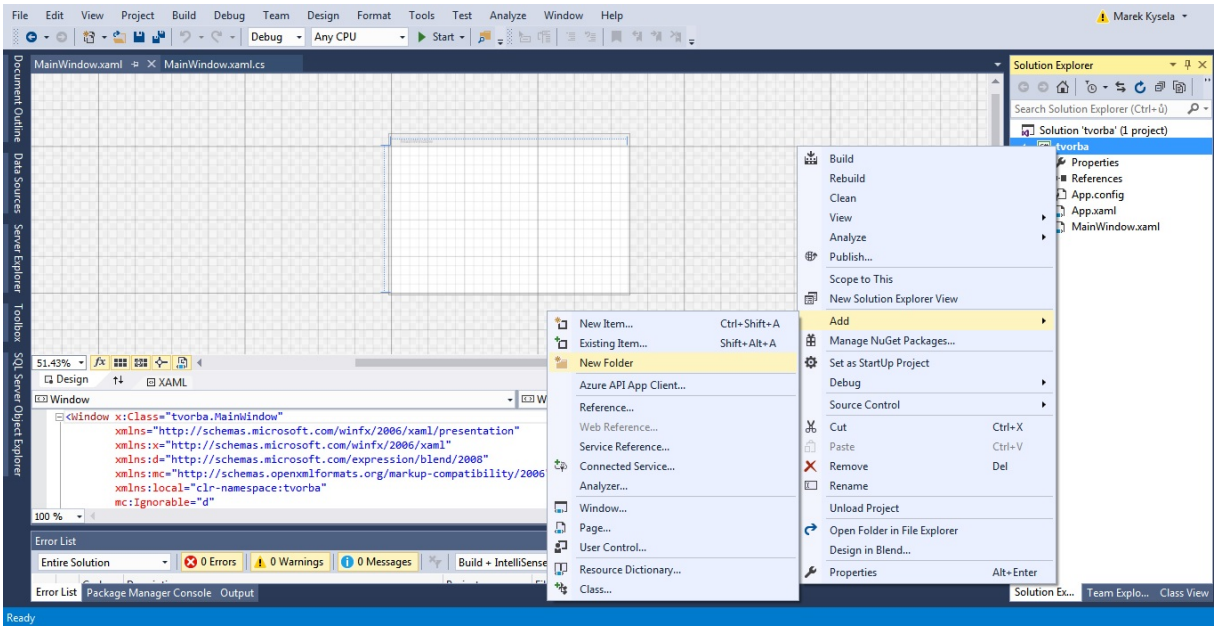
Obrázek 7-8 - Výběr WPF a pojmenování projektu

Po krátké chvíli se objeví prázdné okno MainWindow, jednotlivé složky projektu jsou vidět v levém panelu Solution Explorer. (obrázek 7-9)



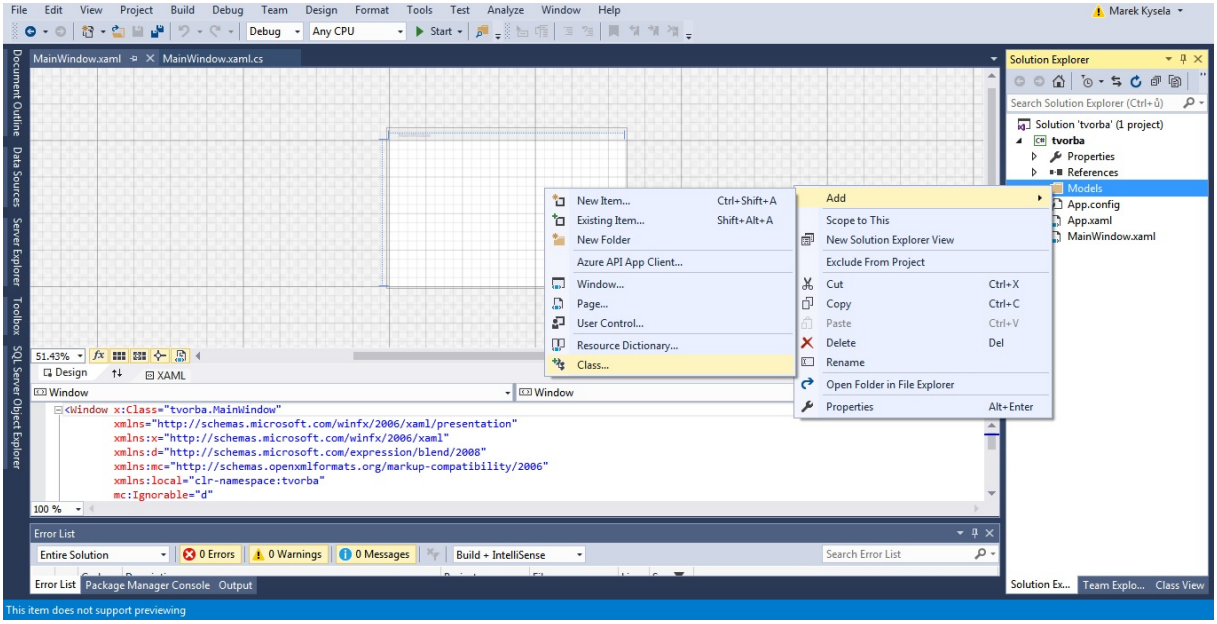
Obrázek 7-9 - Prázdné okno a Solution Explorer

Pro lepší přehlednost Solution Exploreru je nejdříve vytvořena složka Models, do které budou následně vkládány vytvořené třídy. (obrázek 7-10).

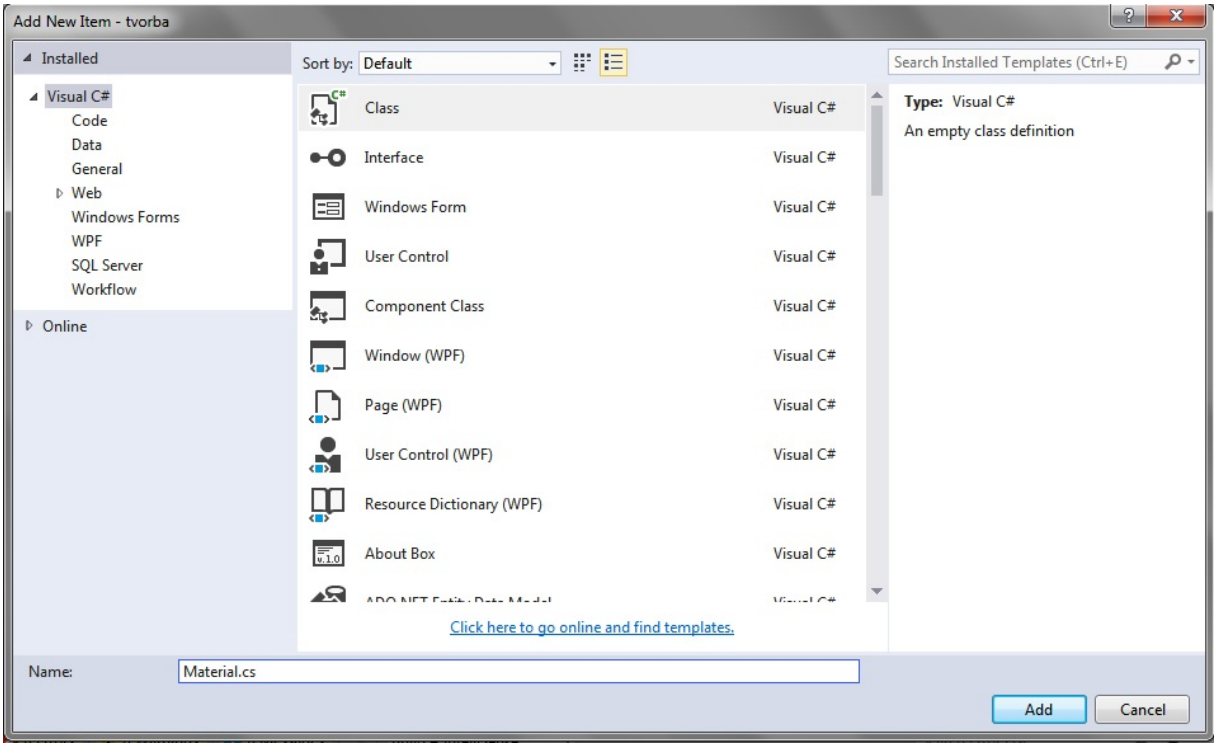


Obrázek 7-10 - Vytvoření složky Models

Dalším krokem je vytvoření tříd. Tyto třídy vycházejí z datové analýzy zvolené aplikace, která byla provedena v předchozí podkapitole. Na obrázcích 7-11 a 7-12 je vidět postup vytvoření třídy.

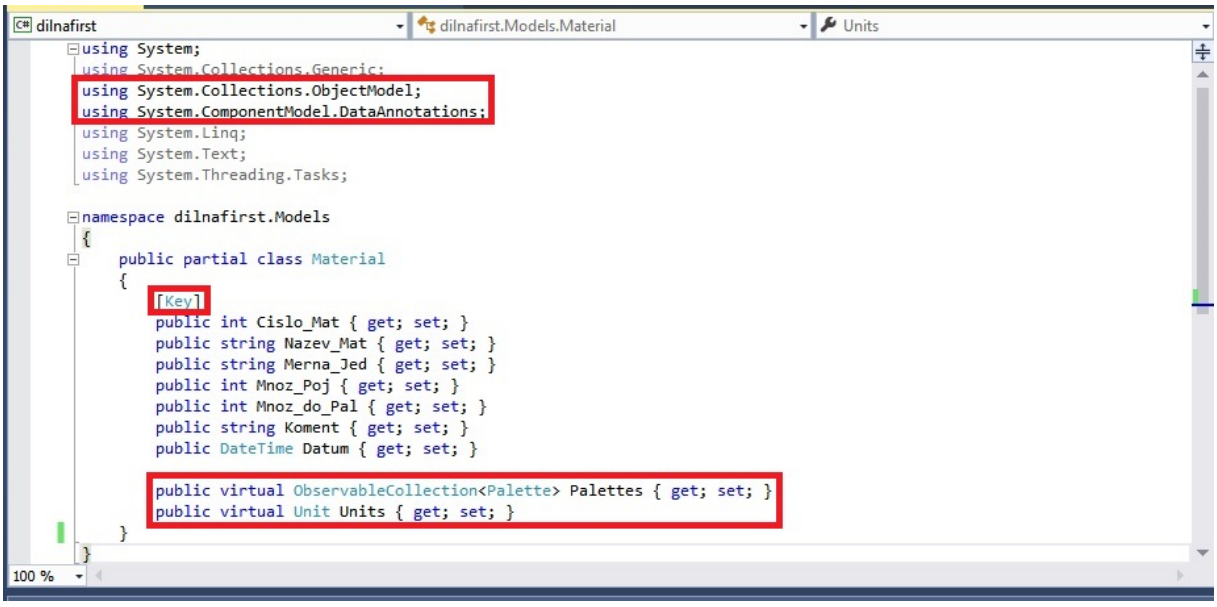


Obrázek 7-11 - Vytvoření nové třídy



Obrázek 7-12 - Pojmenování nové třídy

Nyní je potřeba vytvořenou třídu definovat kódem. Každý atribut se definuje klíčovým slovem public a následně datovým typem a názvem. Dále je potřeba definovat primární klíč key a také cizí klíče. U třídy Materials je cizí klíčem měrná jednotka (Merna_Jed) z tabulky Units. Zároveň je atribut číslo materiálu (Cislo_Mat) cizím klíčem třídy Palette. (obrázek 7-13)



Obrázek 7-13 - Třída Material

Další vytvořenou třídou je třída Palette, primárním klíčem je číslo palety (Cislo_PA), cizími klíči jsou atributy typ palety (Typ_Pa ze třídy Palette_type), stav palety (Stav_Pa ze třídy Palette_state) a číslo materiálu (Cislo_Mat ze třídy Material). (obrázek 7-14)

```
namespace dilnafirst.Models
{
    public partial class Palette
    {
        [Key]
        public int Cislo_PA { get; set; }
        public string Typ_Pa { get; set; }
        public string Stav_Pa { get; set; }
        public string Adr_Ulo { get; set; }
        public int Cislo_Mat { get; set; }
        public int Mnoz_Pa { get; set; }
        public string Koment { get; set; }


        public virtual Material Materials { get; set; }
        public virtual Palette_state Palette_states { get; set; }
        public virtual Palette_type Palette_types { get; set; }

    }
}
```

Obrázek 7-14 - Třída Palette

Na dalších obrázcích jsou vidět třídy Palette_state, Palette_type a Unit, které definuje měrné jednotky pro třídu Material.

```
{
    public partial class Palette_state
    {
        [Key]
        public string Stav_Pa { get; set; }
        public string Popis { get; set; }

        public virtual ObservableCollection<Palette> Palettes { get; set; }
    }
}
```

Obrázek 7-15 - Třída Palette_state

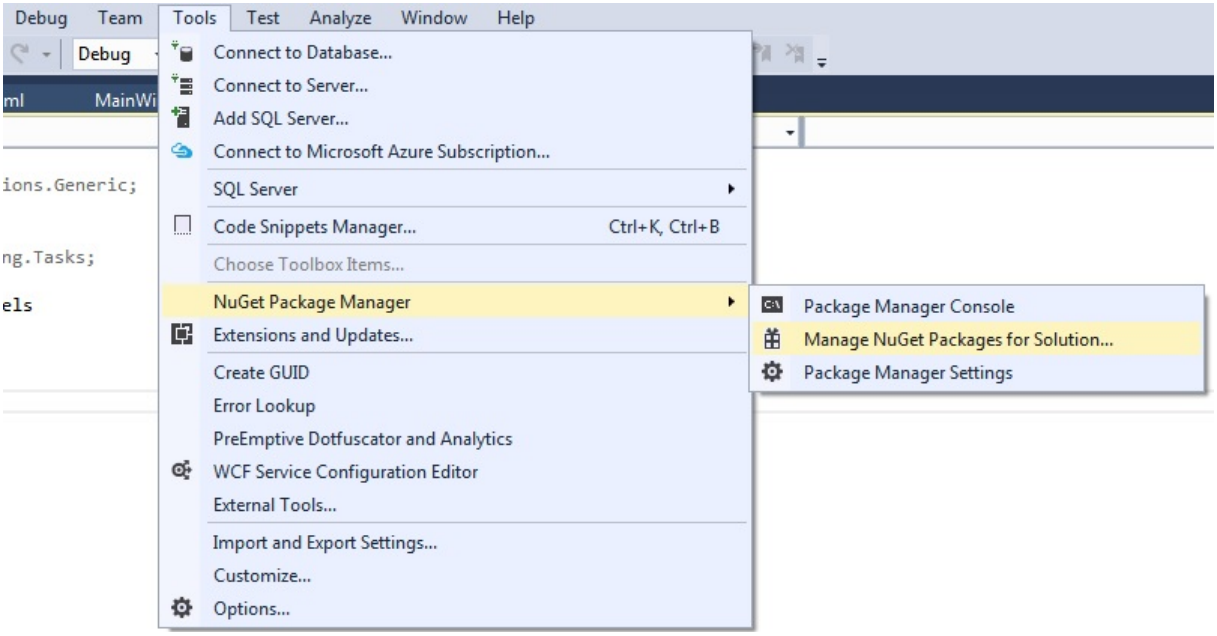
```
{
    public partial class Palette_type
    {
        [Key]
        public string Typ_Pa { get; set; }
        public string Popis { get; set; }

        public virtual ObservableCollection<Palette> Palettes { get; set; }
    }
}
```

Obrázek 7-16 - Třída Palette_type

```
{
    public partial class Unit
    {
        [Key]
        public string Merna_Jed { get; set; }
        public string Popis { get; set; }

        public virtual ObservableCollection<Material> Materials { get; set; }
    }
}
```

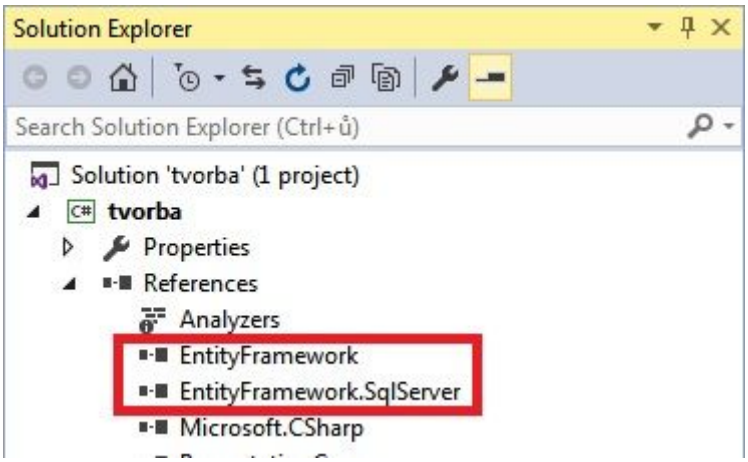
Obrázek 7-17 - Třída Unit

Aby bylo možno vytvořit databázi přístupem Code First, je potřeba přidat do projektu rozšíření entity framework. (obrázky 7-18 a 7-19). Díky tomu se do položky References Solution Exploreru přidají dvě nástavby umožňující pracovat principem Code First. (obrázek 7-20)



Obrázek 7-18 - Přidání Nugetu

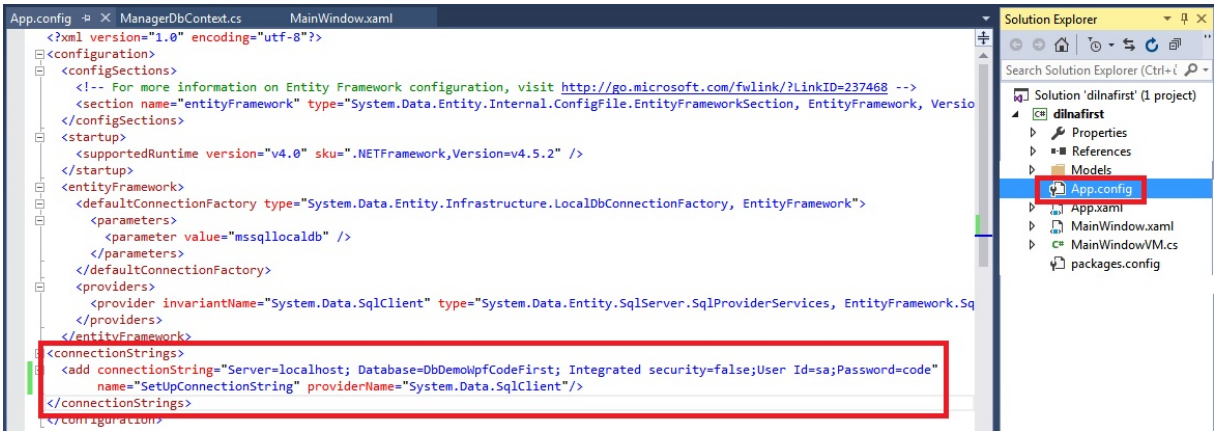
Obrázek 7-19 - Nuget EntityFramework



Obrázek 7-20 – Přidané nástavby

V dalším kroku je potřeba nastavit connection string mezi vytvářenou aplikací a serverem, na kterém bude vytvořena databáze. Tento connection string se ve formě kódu zapíše do položky App.config. V connection stringu se definuje server, na kterém bude vytvořena databáze. Vytvářená databáze se pojmenuje a definuje se uživatel, který se k této databázi bude moci

připojit a bude s ní moci pracovat. Nakonec se pojmenuje vlastní connection string a poskytovatel. (obrázek 7-21)



Obrázek 7-21 - Connection string

V dalším kroku je definována vlastní databáze. Pro tento krok je vytvořena samostatná třída ManagerDbContext. Do této třídy je zapsán výše vytvořený connection string a následně se z dříve definovaných třídy vytvoří tabulky. (obrázek 7-22)

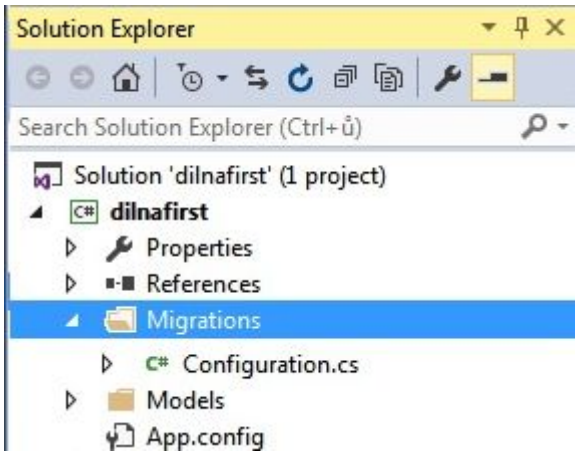
```
namespace dilnafirst.Models


{
    public partial class ManagerDbContext : DbContext
    {
        public ManagerDbContext () : base("name = SetUpConnectionString")
            {

            }

        public DbSet<Material> Materials { get; set; }
        public DbSet<Palette> Palettes { get; set; }
        public DbSet<Unit> Units { get; set; }
        public DbSet<Palette_state> Palette_states { get; set; }
        public DbSet<Palette_type> Palette_types { get; set; }
    }
}
```

Obrázek 7-22 – ManagerDbContext

Pokud chceme aplikaci umožnit komunikaci se serverem, na kterém je vytvořena databáze, musíme v Package Manager Console migrace povolit a to příkazem Enable-Migration. Následně je tento příkaz potvrzen enterem. (obrázek 7-23)

Obrázek 7-23 - Příkaz Enable-Migration

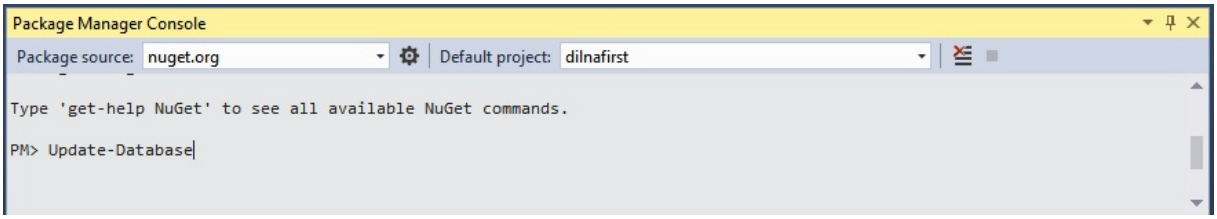
Po provedení tohoto příkazu je v Solution Exploreru automaticky vytvořena složka Migrations. V této složce je vytvořena třída Configuration. (obrázek 7-24) Do této třídy můžeme zadat údaje, které chceme vložit do databáze. Vhodné je v této třídě definovat statická data, například číselníky. (obrázek 7-25) Následně je potřeba provést update databáze. K tomu opět poslouží Package Manager Console, tentokrát s příkazem Update-Database. (obrázek 7-26)



Obrázek 7-24 - Vytvořená složka Migrations a třída Configuration

```
context.Units.AddOrUpdate(
    u => u.Merna_Jed,
    new Unit { Merna_Jed="Kg", Popis="Kilogram"},
    new Unit { Merna_Jed = "t", Popis = "Tuna" },
    new Unit { Merna_Jed = "Ks", Popis = "Kus" },
    new Unit { Merna_Jed = "l", Popis = "Litr" }
    );

context.Palette_states.AddOrUpdate(
    s => s.Stav_Pa,
    new Palette_state { Stav_Pa = "O", Popis = "Paleta je obsazená" },
    new Palette_state { Stav_Pa = "V", Popis = "Paleta je volná" }
    );

context.Palette_types.AddOrUpdate(
    t => t.Typ_Pa,
    new Palette_type { Typ_Pa = "M", Popis = "Malá paleta" },
    new Palette_type { Typ_Pa = "V", Popis = "Velká paleta" }
    );
```

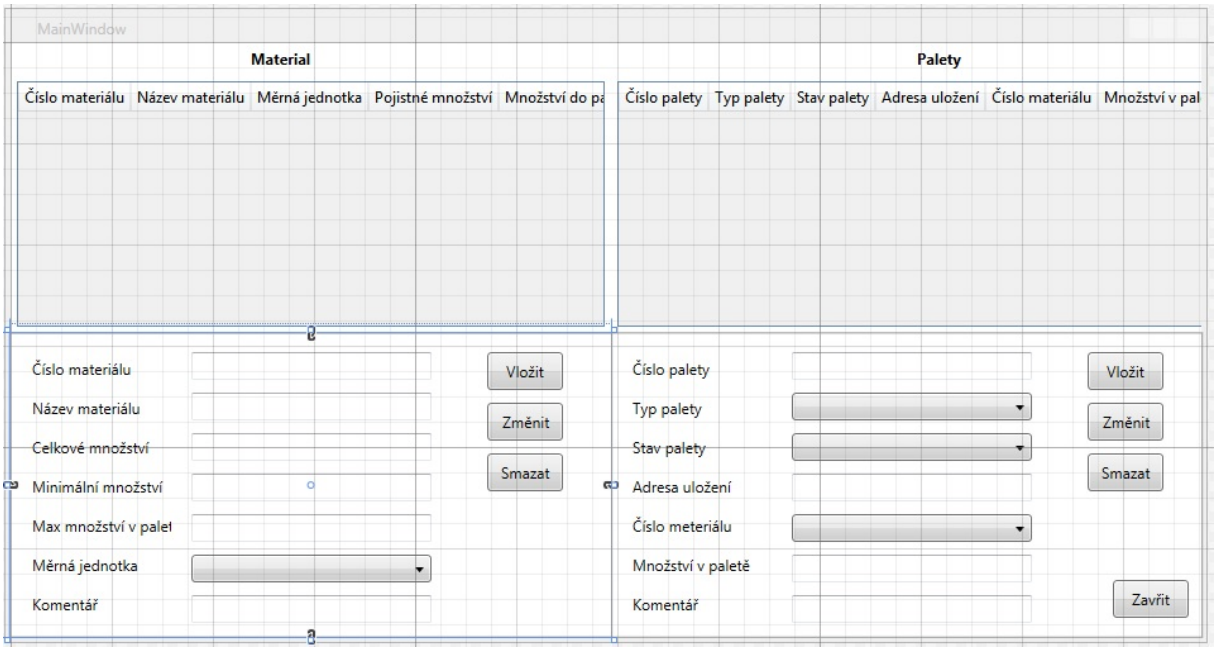
Obrázek 7-25 - Zadání hodnot do číselníků pomocí kódu



Obrázek 7-26 - Příkaz Update-Database

V dalším kroku je proveden návrh jednotlivých komponent formuláře. (obrázek 7-27) V tomto případě byly všechny komponenty spolu s jejich vlastnostmi definovány pomocí kódu. Na obrázku 7-28 jsou definovány základní rozměry formuláře rozdělení tohoto

formuláře na hlavní části. Na obrázku 7-29 je definován datagrid pro tabulku (třídu) materiál. V této části kódu je patrné provázání databáze s aplikací pomocí klíčového slova binding (vazba). Nastavení módu na twoway znamená, že uživatel může data jak prohlížet, tak i upravovat. Na obrázku 7-30 jsou v kódu definována tlačítka a další komponenty formuláře. Atribut Click odkazuje na událost při kliknutí na tlačítko. Událost Zavrit_Click tlačítka zavřít je definována na obrázku 7-31. Pro zobrazení dat na formuláři je podstatné definovat kontext, který se má v daných polích zobrazit. (obrázek 7-32)



Obrázek 7-27 - Návrh formuláře

```
   Title="MainWindow" Height="500" Width="950">
Grid>
   <Grid.RowDefinitions>
       <RowDefinition Height="Auto"/>
       <RowDefinition Height="*" />
       <RowDefinition Height="240"/>
   </Grid.RowDefinitions>

   <!--Row 1-->
   <Label Grid.Row="0" Content="Material" HorizontalAlignment="Left" FontWeight="Black" Margin="185,0,0,0"/>
   <Label Grid.Row="0" Content="Palety" HorizontalAlignment="Right" FontWeight="Black" Margin="0,0,185,0"/>

   <!--Row 2-->
   <Grid Grid.Row="1">

       <Grid.ColumnDefinitions>
           <ColumnDefinition Width="475"/>
           <ColumnDefinition Width="475"/>

       </Grid.ColumnDefinitions>
```

Obrázek 7-28 - Základní parametry formuláře

```
<DataGrid Grid.Column="0" Name="DataGridMaterial" Margin="5,5,5,5"
      RenderOptions.ClearTypeHint = "Enabled"
      TextOptions.TextFormattingMode="Display"
      CanUserAddRows="False"
      CanUserDeleteRows="False"
      SelectionUnit="FullRow"
      ItemsSource="{Binding Materials, Mode=TwoWay}"
      AutoGenerateColumns="False">
    <DataGrid.Columns>
        <!--Column 1-->
        <DataGridTextColumn Header="Číslo materiálu" Binding="{Binding Cislo_Mat}"/>
        <!--Column 2-->
        <DataGridTextColumn Header="Název materiálu" Binding="{Binding Nazev_Mat}"/>
        <!--Column 3-->
        <DataGridComboBoxColumn Header="Měrná jednotka" ItemsSource="{Binding DataContext.Materials,
                                RelativeSource ={RelativeSource AncestorType={x:Type Window}}}"
                                SelectedValuePath="Merna_Jed" DisplayMemberPath="Merna_Jed"/>

        <!--Column 4-->
        <DataGridTextColumn Header="Pojistné množství" Binding="{Binding Mnoz_Poj}"/>

        <!--Column 5-->
        <DataGridTextColumn Header="Množství do palety" Binding="{Binding Mnoz_do_Pal}"/>

        <!--Column 6-->
        <DataGridTextColumn Header="Komentář" Binding="{Binding Koment}"/>
```

Obrázek 7-29 - Definování datagridu

```
<Grid Grid.Column="1">
    <StackPanel HorizontalAlignment="Center" Width="100" Orientation="Vertical" Margin="350,10,0,0">
        <Button Name="VlozitPal" Content="Vložit" Width="60" Height="30" Margin="5,5,15,5" Click="VlozitPal_Click"/>
        <Button Name="ZmenitPal" Content="Změnit" Width="60" Height="30" Margin="5,5,15,5" Click="ZmenitPal_Click"/>
        <Button Name="SmazatPal" Content="Smazat" Width="60" Height="30" Margin="5,5,15,5" Click="SmazatPal_Click"/>
        <Button Name="Zavrit" Content="Zavřít" Width="60" Height="30" Margin="35,65,5,5" Click="Zavrit_Click"/>
    </StackPanel>
    <StackPanel Width="120" Margin="-300,10,30,0">
        <Label  Content="Číslo palety" Margin="0,5,0,0"/>
        <Label  Content="Typ palety" Margin="0,5,0,0"/>
        <Label  Content="Stav palety" Margin="0,5,0,0"/>
        <Label  Content="Adresa uložení" Margin="0,5,0,0"/>
        <Label  Content="Číslo meteriálu" Margin="0,5,0,0"/>
        <Label  Content="Množství v paletě" Margin="0,5,0,0"/>
        <Label  Content="Komentář" Margin="0,5,0,0"/>
    </StackPanel>
    <StackPanel Width="200" Margin="0,10,0,0">
        <TextBox  Margin="5,5,5,5" Text="{Binding CurrentSelectedPalette.Cislo_PA}"/>
        <ComboBox Margin="5,5,5,5"
              ItemsSource="{Binding Datacontext.Palette_types}"
              SelectedValuePath="Merna_Jed"
              DisplayMemberPath="Merna_Jed"
              SelectedValue="{Binding Datacontext.CurrentSelectedPallete.Typ_Pa}"
              />
        <ComboBox Margin="5,5,5,5"
              ItemsSource="{Binding Datacontext.Palette_states}"
              SelectedValuePath="Merna_Jed"
              DisplayMemberPath="Merna_Jed"
              SelectedValue="{Binding Datacontext.CurrentSelectedPallete.Stav_Pa}"
```

Obrázek 7-30 - Definování tlačítek

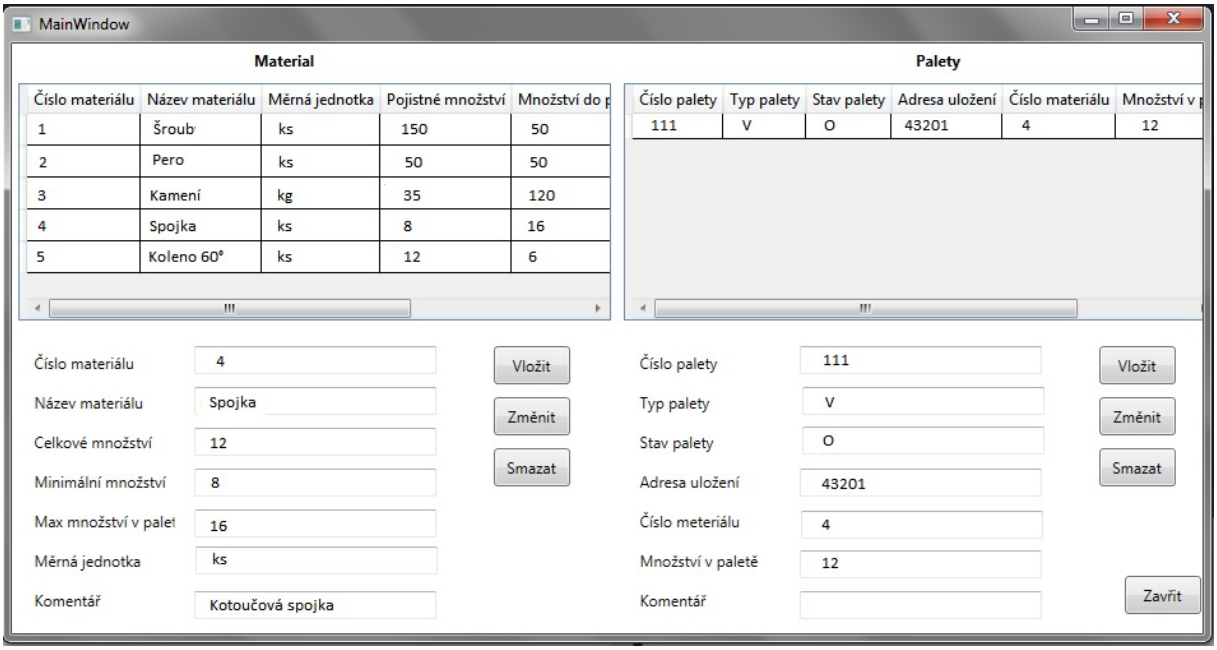
```
private void Zavrit_Click(object sender, RoutedEventArgs e)
{
    Close();
}
```

Obrázek 7-31 - Událost pro tlačítko Zavřít

```
private ObservableCollection<Material> _materials;
public ObservableCollection<Material> Materials
{
    get {
        if (_materials==null)
        {
            _materials= new ObservableCollection<Material>(_context.Materials.Include(m => m.Units));
        }
        return _materials;

    }
    set
    {
        _materials = value;
        OnPropertyChanged("Materials");
    }
}
```

Obrázek 7-32 - Definování kontextu

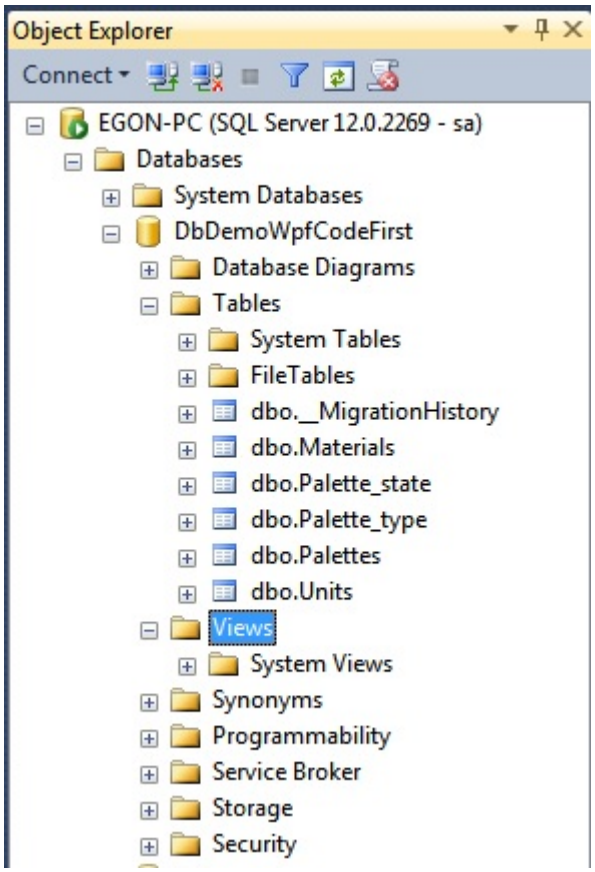
Na obrázku 7-33 je vidět výsledná aplikace. Vlevo je přehled materiálů, vpravo je přehled palet. Ve spodní části formuláře jsou komponenty pro změnu a zadávání dat.



Obrázek 7-33 - Vytvořená aplikace

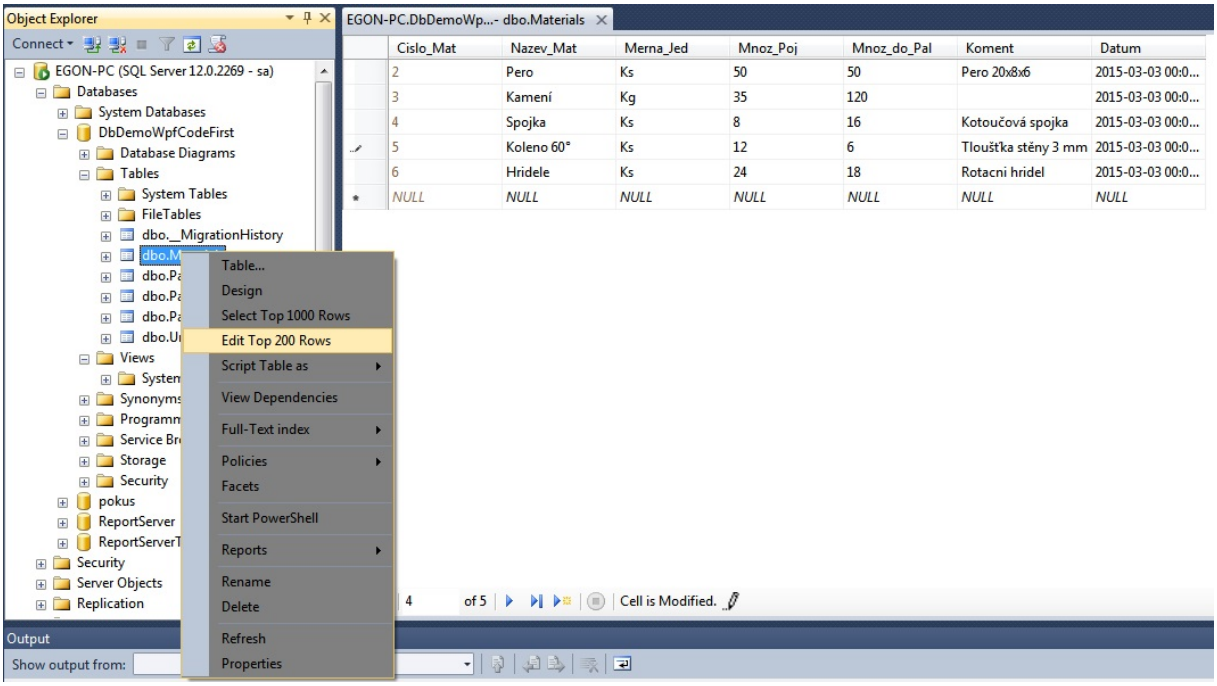
7.4 Práce s aplikací na serveru

Po spuštění SQL Server Management Studia lze najít vytvořenou databázi v Object Exploreru. (obrázek 7-34) Kromě samotných tabulek, které tvoří databázi, je na serveru také tabulka migrací, která se vytvořila automaticky. Jedná se o přehled všech migrací, které byly spuštěny.



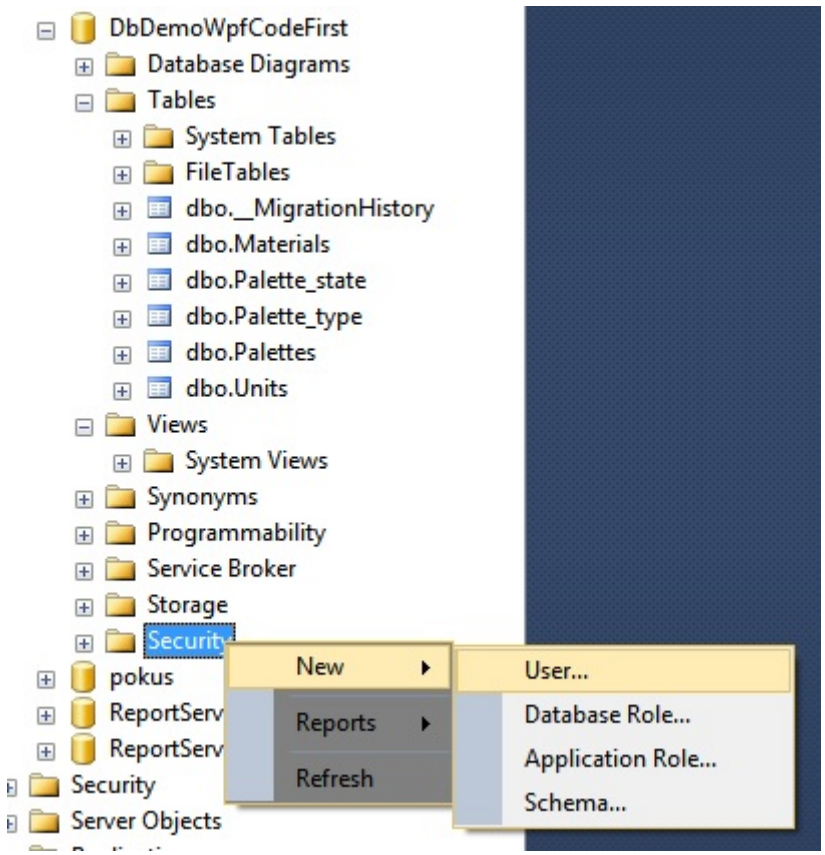
Obrázek 7-34 - Vytvořená databáze

Management Studio umožňuje uživateli práci s daty i bez samotné aplikace. Data lze prohlížet, přidávat, měnit nebo mazat. (obrázek 7-35) Tyto možnosti jsou závislé na oprávnění uživatele.

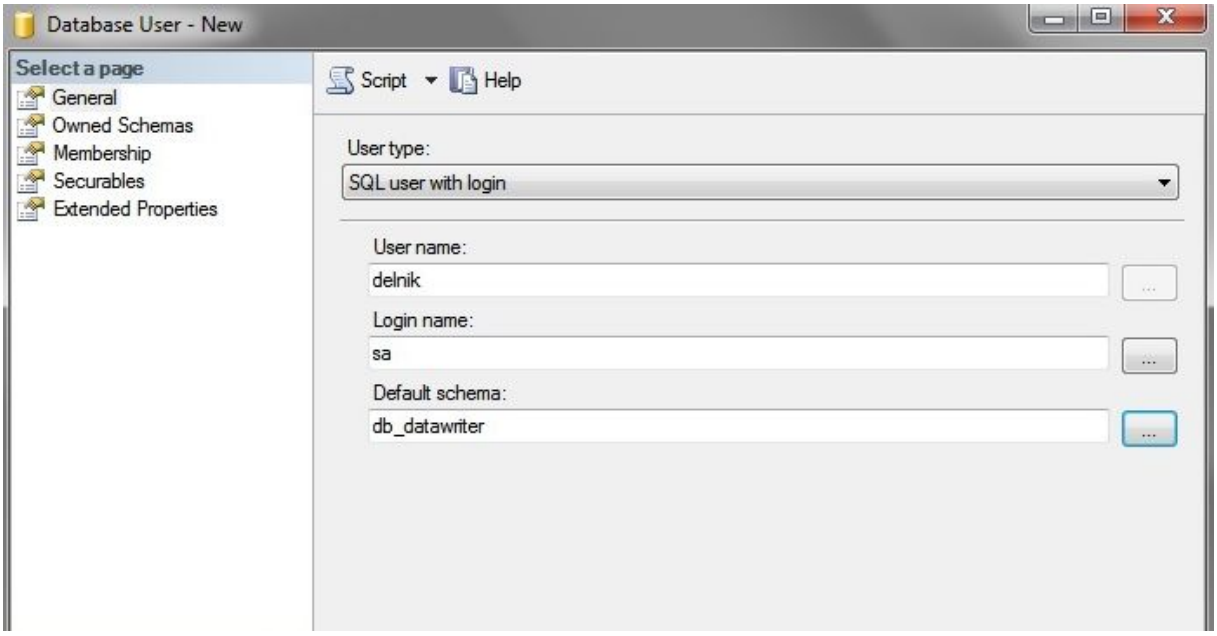


Obrázek 7-35 - Úprava dat na serveru

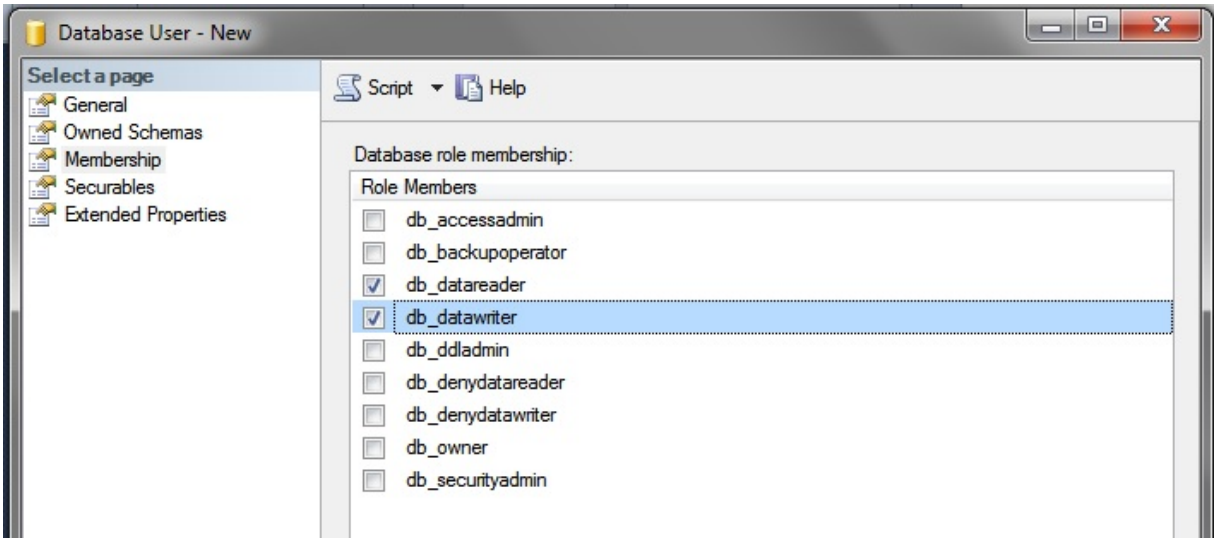
Kromě práce s vlastní databází lze v Management Studiu například přidávat nové uživatele. Činí se tak přes složku Security (pozor na složku Security pro celý server). Postup přidání nového uživatele je vidět na obrázcích 7-36 až 7-39.



Obrázek 7-36 - Přidání nového uživatele



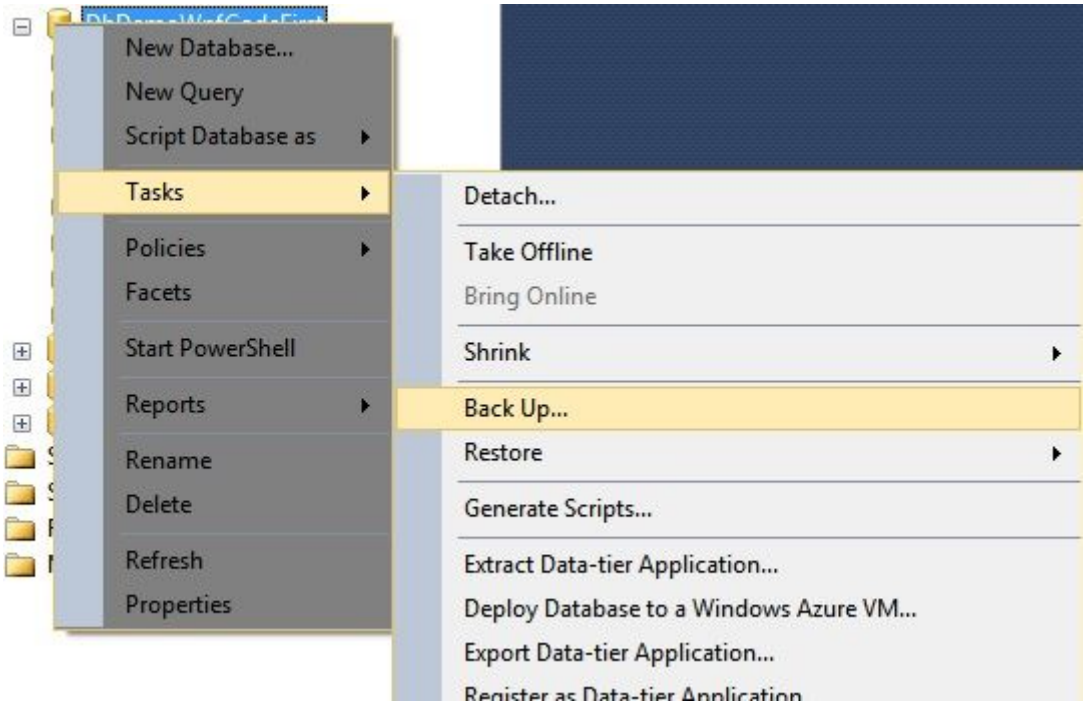
Obrázek 7-37 - Základní údaje nového uživatele



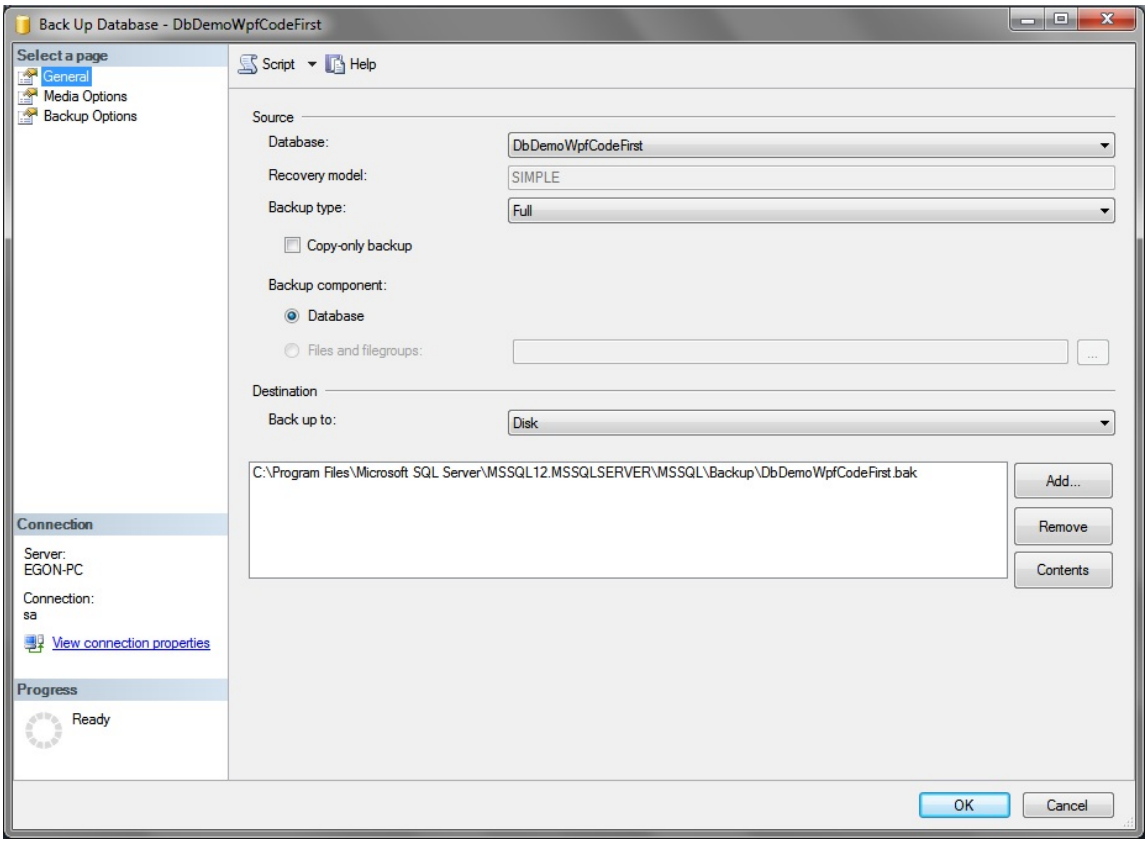
Obrázek 7-38 - Definování rolí uživatele

Obrázek 7-39 - Nový uživatel ve složce Security

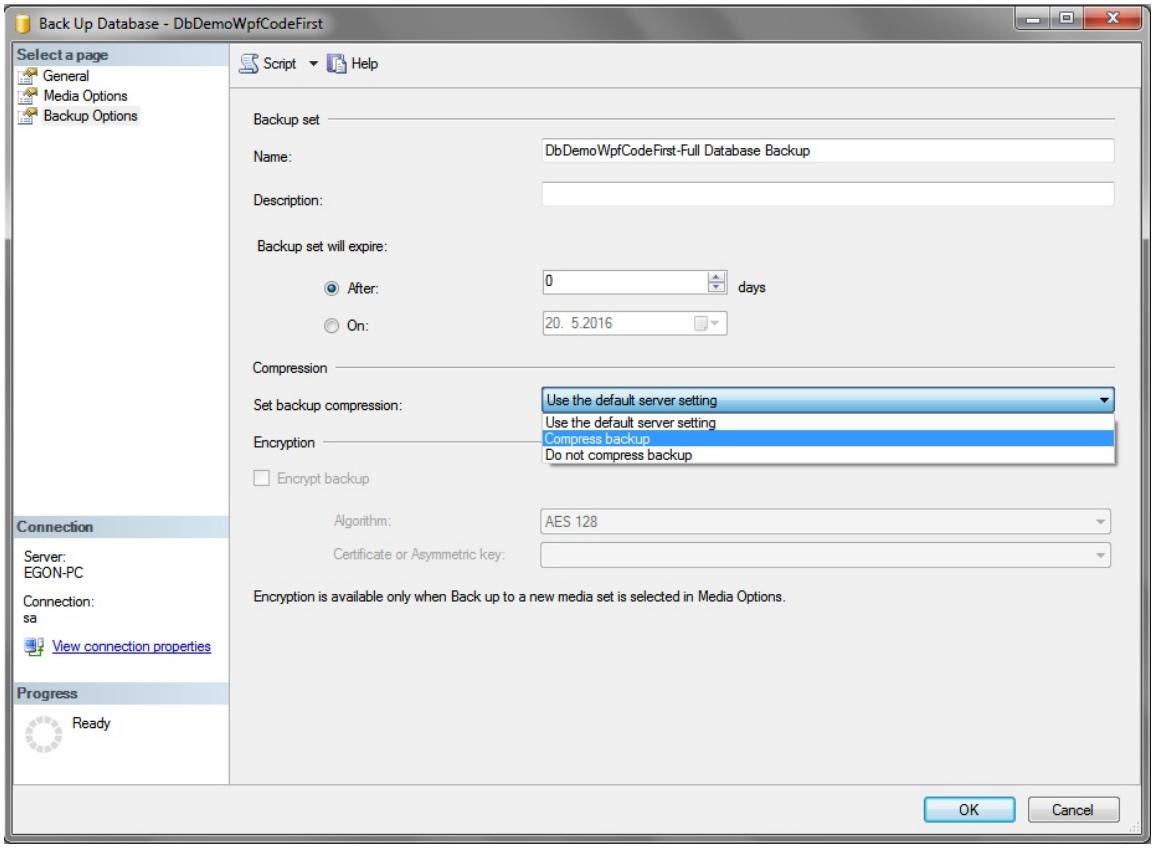
Důležitým bodem je nastavení zálohování. (obrázek 7-40) U toho lze zvolit úplnou nebo rozdílovou zálohu. Dále se nastaví zálohovací destinace. (obrázek 7-41) V dalším nastavení lze zvolit kompresi záloh a také dobu, po jejímž uplynutí se zálohy vymažou. (obrázek 7-42)



Obrázek 7-40 - Položka zálohování v možnostech serveru

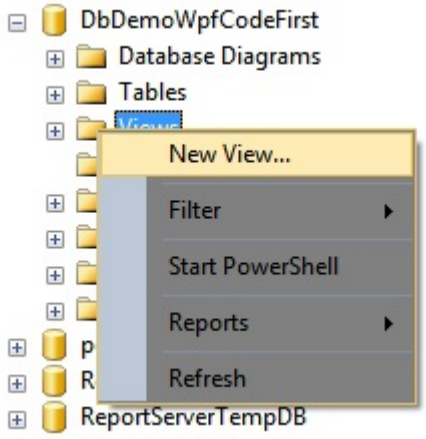


Obrázek 7-41 - Obecné nastavení zálohování



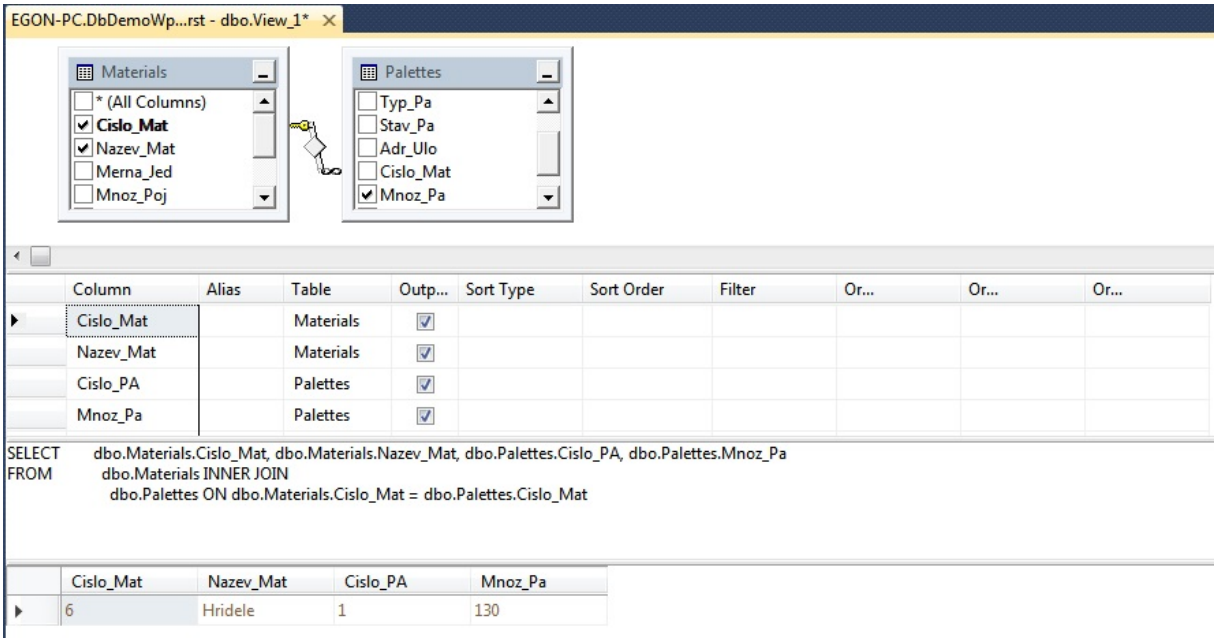
Obrázek 7-42 - Možnost komprese záloh

Další výhodou práce na serveru je, že si můžeme snadno vytvářet vlastní pohledy (views) z dané databáze. (obrázek 7-43)



Obrázek 7-43 - Vytvoření pohledu

Na obrázku 7-44 je vidět vytvořený pohledy, který má atributy číslo materiálu a název z tabulky materiálů a dále atributy číslo palety a z tabulky palet. Pohled je realizován SQL dotazem.



Obrázek 7-44 - Vytvořený pohled

8 Závěr

V mé diplomové práci jsou v úvodu popsány základní pojmy vybraného tématu. Následně se práce věnuje popisu jednotlivých komponent MS SQL Serveru. Poslední část práce se věnuje vlastní implementaci MS SQL Serveru pro vytvořenou databázi. Důraz je kladen především na zabezpečení serveru, to je ošetřeno pomocí šifrování a především pomocí nastavení rolí (serverových a databázových) u uživatelů SQL Serveru. Tyto role jsou podrobně vysvětleny a je tak patrné, jaká práva mají uživatelé s určitými rolemi. Dalším důležitým krokem při implementaci MS SQL Serveru je vybrání a tvorba plánu zálohování a obnovy. V práci jsou popsány jednotlivé typy záloh a modely obnovení. Dále jsou popsány jednotlivá zálohovací zařízení a média. Nakonec jsou vypsány jednotlivé zálohovací strategie. Vlastní implementace je realizována na aplikaci vytvořené pomocí platformy WPF a nástavbou Entity Framework.

Žádný strojírenský podnik nelze efektivně řídit bez aktuálních a přesných dat. Tato data zároveň potřebujeme v reálném čase i přesto, že se mohou neustále měnit. Toho lze dosáhnout právě díky softwaru MS SQL Server, kdy se k databázi můžeme při správném nastavení připojit kdykoliv a odkudkoliv. Zároveň lze data na serveru zabezpečit lépe než na lokální stanici.

Myslím, že by se implementace serveru mohla projevit i v předmětu DBC na naší katedře, neboť tvorba aplikace, která pracuje s databází bez připojení k serveru, nemá v dnešní době význam. Má práce by mohla posloužit alespoň jako částečný návod, jak server zprovoznit a co vše důležitého na něm nastavit.

Seznam obrázků

Seznam zdrojů a použité literatury

[1] Wiley, John & Sons., *Windows Server Administration Fundamentals*. Hoboken: Microsoft Official Academic Course, 2011. ISBN 978-0-470-90182-3.

[2] Holub, V., Kopeček, P., *Databázové systémy ve strojírenství – Přednášky*, ebook. Plzeň: Západočeská univerzita v Plzni, 2010.

[3] Comer, Douglas E., David L. Stevens., *Internetworking with TCP/IP, Vol. III: Client-Server Programming and Applications*. West Lafayette: Department of Computer Sciences, Purdue University, 1993. ISBN 0-13-474222-2.

[4] Lacko, Ľuboslav, *1001 tipů a triků pro SQL*. Brno: Computer Press, 2011. ISBN 978-80-251-3010-0.

[5] Riordan, Rebecca M., *Vytváříme relační databázové aplikace*. Praha: Computer Press

2000. ISBN: 8072263609.

[6] Pokorný, Jan, *Učíme se SQL*. Praha: Plus, 1993. ISBN 80-85297-47-7

[7] Delaney, Kalen, *Inside Microsoft SQL Server 2000*. Microsoft Press, a division of Microsoft Corporation, 2001. ISBN 0-7356-0998-5.

[8] LeBlanc, Patrick, *Microsoft SQL Server 2012 Step by Step*. Microsoft Press, 2013. ISBN 978-0-7356-6390-9.

[9] Stanek, William. R., *Microsoft SQL Server 2012 Kapesní rádce administrátora.* Brno: Computer Press, 2013. ISBN 978-80-251-3797-0.

[10] Walters, Robert E., *Mistrovství v SQL Server 2008 – Kompletní průvodce databázového experta*. Brno: Computer Press, 2009. ISBN 978-80-251-2329-4.

[11] Solis, Daniel, *Illustrated WPF* [online]. 2009. [cit. 2016-05-19]. ISBN 978-1-4302-1910-

1. Dostupné z: http://fast-file.blogspot.com

[12] Nathan, Adam, *Windows Presentation Foundation Unleashed* [online]. 2006. [cit. 2016-05-19]. ISBN 978-0672328916. Dostupné z: www.it-ebooks.info

[13] what-is-entityframework. Entity Framework Tutorial. [online]. [cit. 2016-05-19]. Dostupné z: http://www.entityframeworktutorial.net/what-is-entityframework.aspx

[14] Petrovic, Milena, *Using custom reports to improve performance reporting in SQL Server 2014 – the basics.* SQLShack. [online]. 8.9.2014 [cit. 2016-05-19]. Dostupné z: http://www.sqlshack.com/using-custom-reports-improve-performance-reporting-sql-server-2014-basics/